Ann Oper Res (2007) 150:231–244
DOI 10.1007/s10479-006-0155-z

A note on the parametric maximum flow problem and some related reoptimization issues

Maria Grazia Scutellà

Published online: 21 December 2006

Abstract In this paper, we will extend the results about the parametric maximum flow problem to networks in which the parametrization of the arc capacities can involve both the source and the sink, as in Gallo, Grigoriadis, and Tarjan (1989), and also an additional node. We will show that the minimum cuts of the investigated networks satisfy a relaxed form of the generalized nesting property (Arai, Ueno, and Kajitani, 1993). A consequence is that the corresponding parametric maximum flow value function has at most $n - 1$ breakpoints. All the minimum cut capacities can therefore be computed by $O(1)$ maximum flow computations.

We will show then that, given $O(n)$ increasing values of the parameter, it is possible to compute the corresponding maximum flows by $O(1)$ maximum flow computations, by suitably extending Goldberg and Tarjan's maximum flow algorithm.

Keywords Maximum flow · Parametric arc capacity

Introduction

The classical maximum flow problem calls for finding a maximum flow (or, equivalently, a minimum cut) in a capacitated network. In several applications, often it is required to solve a sequence of maximum flow problems. In particular, there are situations in which the arc capacities are not fixed, but they are functions of a single parameter, and the goal is to compute a maximum flow for a sequence of $O(n)$ increasing values of the parameter, where n is the cardinality of the set of nodes of the network. Hereafter we will refer to this problem as to the *parametric maximum flow problem*.

Gallo, Grigoriadis, and Tarjan (1989) showed that an important class of the parametric maximum flow problem can be solved via reoptimization techniques, by extending the maximum flow algorithm devised by Goldberg and Tarjan (1988), also known as the *push-relabel* algorithm. The resulting algorithm has a worst-case time bound which is only a constant

M. G. Scutellà (✉)
Dipartimento di Informatica, Largo B. Pontecorvo 3, I-56127 Pisa, Italy
e-mail: scut@di.unipi.it

factor greater than the time bound for the non-parametric problem. The particular parametric maximum flow problem addressed by Gallo, Grigoriadis, and Tarjan (1989) is such that the capacities of the arcs leaving the source node are nondecreasing functions of the parameter, the capacities of the arcs entering the sink node are nonincreasing functions of the parameter, whereas all the other arc capacities are constant. As observed by the authors, the proposed algorithm also applies if the capacities of the arcs leaving the source are nonincreasing functions of the parameter, the capacities of the arcs entering the sink are nondecreasing functions of the parameter, and the values of the parameter are given in a decreasing order. See also (Gusfield and Martel, 1992) for some algorithmic generalizations.

McCormick (1999) extended the results of Gallo, Grigoriadis, and Tarjan (1989) to more general parametric networks. He considered the case in which the subgraph induced by the arcs whose capacity is function of the parameter has the following arborescence-shaped structure, which generalizes the one studied by Gallo, Grigoriadis, and Tarjan (1989). There is an out-arborescence S rooted at the source (i.e., a tree directed away from the source), where the (affine linear) arc capacities are nondecreasing function of the parameter, and there is an in-arborescence rooted at the sink (i.e., a tree directed towards the sink), where the (affine linear) arc capacities are nonincreasing function of the parameter. All the other arc capacities are constant. Furthermore, there is a sort of superadditivity relation linking the coefficients of the linear capacities, which will be formally described in Section 1. Also in this case, the extension of the maximum flow algorithm devised by Gallo, Grigoriadis, and Tarjan (1989) is able to solve the parametric maximum flow problem by $O(1)$ maximum flow computations.

A quite different situation has been addressed by Arai, Ueno, and Kajitani (1993). The authors studied the case in which the set of arcs whose capacity depends on the parameter is the set of arcs incident to a single node u^* (other than the source and the sink of the maximum flow network). Under the hypothesis that all these arc capacities are nondecreasing functions of the parameter, Arai et al. showed that the parametric maximum flow algorithm mentioned above can be used to determine all the minimum cut capacities (and so, all the maximum flow values) by $O(1)$ maximum flow computations. However, the authors did not discuss how to compute the corresponding maximum flows.

The results mentioned above are strictly related to special properties concerning the minimum cuts of the parametric networks. More specifically, the minimum cuts of the parametric networks addressed by Gallo, Grigoriadis, and Tarjan (1989), and by McCormick (1999) satisfy the so-called "nesting" property, whereas the minimum cuts of the parametric networks studied by Arai, Ueno, and Kajitani (1993) satisfy a generalization of the nesting property. These properties will be reviewed in Section 2.1.

In all cases, if one considers the maximum flow value (or, equivalently, the minimum cut capacity) as a function of the parameter, the nesting and the generalized nesting properties imply that the parametric maximum flow value function has at most $n - 1$ breakpoints (under the assumption that the parametric arc capacities are affine linear functions). The extension of the maximum flow algorithm devised by Gallo, Grigoriadis, and Tarjan (1989) allows one to determine these $O(n)$ breakpoints by $O(1)$ maximum flow computations. Such an interesting result may not hold when some arc capacities are nonincreasing functions of the parameter (Carstensen, 1983).

Here we will extend the results about the parametric maximum flow problem to the case of networks in which the parametrization of the arc capacities can involve both the source and the sink, as in Gallo, Grigoriadis, and Tarjan (1989), and also a node u^* other than the source and the sink. More precisely, we will address the situation in which the capacities of the arcs leaving the source are nondecreasing functions of the parameter, and the capacities of the

arcs entering the sink are nonincreasing functions of the parameter. Furthermore, for a node u^* other than the source and the sink, the capacities of the arcs entering u^* are nonincreasing functions of the parameter, whereas the capacities of the arcs leaving u^* are nondecreasing functions of the parameter.

We will show that the minimum cuts of the investigated parametric networks satisfy a relaxed form of the generalized nesting property introduced by Arai, Ueno, and Kajitani (1993). A consequence of this property is that, also in this kind of networks, the parametric maximum flow value function has at most $n - 1$ breakpoints. By the same arguments of Arai et al., the parametric maximum flow algorithm of Gallo, Grigoriadis, and Tarjan (1989) can therefore be used to determine all the minimum cut capacities (and so, all the maximum flow values) by $O(1)$ maximum flow computations.

Then, the results of Arai, Ueno, and Kajitani (1993) will be enhanced showing that, given a sequence of $O(n)$ increasing values of the parameter, for the studied kind of parametric networks it is possible to compute the corresponding maximum flows in $O(1)$ maximum flow computations, by suitably extending Goldberg and Tarjan's maximum flow algorithm. Furthermore, we will show that a minor modification of the proposed algorithm is able to solve the parametric maximum flow problem, with the same time complexity, for the parametric networks of Arai, Ueno, and Kajitani (1993), so enforcing their result.

We want to emphasize that the studied parametric maximum flow problem fits into a very general family of parametric problems, described by Brumelle, Granot, and Liu (2002), which comprises both the cases investigated by Gallo, Grigoriadis, and Tarjan (1989) and by Arai, Ueno, and Kajitani (1993), and, partially, the case studied by McCormick (1999). For this family, necessary and sufficient conditions have been stated for the existence of a "totally ordered selection" of minimum cuts, that is, the existence of minimum cuts which are totally ordered with respect to the partial order $\ll$ so defined: given two minimum cuts (X_p, Y_p) and (X_q, Y_q), $(X_p, Y_p) \ll (X_q, Y_q)$ if $X_p \subseteq X_q$. Whereas the parametric networks of Arai, Ueno, and Kajitani (1993) can be proved to satisfy the necessary and sufficient conditions devised by Brumelle, Granot and Liu (2002) (that is, the generalized nesting property derives from such conditions), it is not possible to exploit such conditions in order to decide, in polynomial time, whether the parametric networks studied in this work do possess a totally ordered selection of minimum cuts. Therefore, since the relaxed form of the generalized nesting property, proved in this work, is not a special case of these conditions (it does not imply the existence of a totally ordered selection of minimum cuts, although it implies the existence of $O(n)$ breakpoints for the parametric maximum flow value function), the results of this note concerning minimum cuts can not be interpreted as a special case of known results from the literature, and they have to be considered as original results.

The plan of the paper is the following. In Section 1 we will review the push-relabel algorithm and the parametric maximum flow algorithms proposed in the literature sofar. Then, in Section 2 we will describe the new parametric networks under investigation and the related results.

1 The parametric maximum flow problem

1.1 The push-relabel algorithm

Let $G = (V, E)$ be a directed graph, with $|V| = n$ and $|E| = m$, having a distinguished source node s and a distinguished sink node t. Let $c(v, w)$ be the nonnegative capacity associated with (v, w), $\forall (v, w) \in E$. A *flow* on G is a real-valued function on the arcs which satisfies

the capacity constraints and the flow conservation constraints. The *value* of a flow is the net flow entering the sink. A *maximum flow* is a flow of maximum value.

A *cut* (X, Y) is a node partition such that $s \in X$ and $t \in Y$, and its capacity is $c(X, Y) = \sum_{v \in X, w \in Y} c(v, w)$. A *minimum cut* is a cut of minimum capacity. The *max-flow min-cut theorem* of Ford and Fulkerson (1962) states that the maximum flow value is equal to the minimum cut capacity.

The push-relabel algorithm of Goldberg and Tarjan maintains a flow relaxation called "preflow". A *preflow* f on G is a real-valued function on the arcs which satisfies the capacity constraints and a relaxation of the flow conservation constraints, stating that the total flow entering a node v other than s can be greater than or equal to the outgoing flow. Each node $v \neq s$ can therefore have an *excess of flow* $e(v) = \sum_{(w,v)} f(w, v) - \sum_{(v,w)} f(v, w)$. If $e(v) > 0$, v is said to be an *active node*. The push-relabel algorithm starts with the preflow f which saturates all the arcs leaving s and holds zero for all the other arcs, and tries to push the excesses from the active nodes toward the sink along the residual graph associated with f, in such a way to transform the preflow into a maximum flow.

In order to push the flow, the algorithm maintains a *valid distance label* $d(v)$ for each node v, where a valid distance d is a function on the nodes of G such that $d(t) = 0$, $d(s) = n$ and $d(v) \leq d(w) + 1$ for each residual arc (v, w), i.e., each arc of the residual graph associated with f ((v, w) is a residual arc if either it is an arc of G such that $f(v, w) < c(v, w)$, or (w, v) is arc of G such that $f(w, v) > 0$; in the first case the residual capacity of (v, w) is $r(v, w) = c(v, w) - f(v, w)$, while in the latter case it is $r(v, w) = f(w, v)$).

As proved by Goldberg and Tarjan (1988), the distance label $d(v)$ estimates, from below, the length of the shortest augmenting paths (i.e., the shortest paths in the residual graph) from v to t. As a consequence, $d(v) \geq n$ implies that no augmenting path exists from v to t. Indeed, it is immediate to prove that $d(s)$ can be set to any value greater than or equal to n in order to maintain all the properties of the valid distance functions. Hereafter we will therefore consider such a more general definition of valid distance function.

In detail, in its more general form the push-relabel algorithm consists of selecting an active node v and applying the appropriate operation, until there are no active nodes in G:

- (*push*) if there is a residual arc (v, w) such that $d(v) = d(w) + 1$, then send a flow $\delta = min\{e(v), r(v, w)\}$ along (v, w), and update $e(v)$ and $e(w)$;
- (*relabel*) if there is no residual arc (v, w) such that $d(v) = d(w) + 1$, then increase the distance label $d(v)$ to $min\{d(w) + 1 : (v, w) \text{ is a residual arc}\}$.

Observe that the distance labels increase as the flow excesses are pushed toward the sink; specifically, they increase at most to the value $d(s) + n$ (Goldberg and Tarjan, 1988).

Goldberg and Tarjan proved that their algorithm is correct; that is, when it terminates, f is a maximum flow. Moreover, at termination a minimum cut can be easily obtained by setting $d(v)$ to the length of the shortest augmenting paths from v to t, for each node v for which an augmenting path exists, and setting $d(v) = n$ if the current label is less than n and no augmenting path from v to t exists. This process can be performed via a breadth-first visit of the residual graph starting from t, in $O(m)$ time. Then, a minimum cut (X, Y) can be computed by inserting to X each node v such that $d(v) \geq n$, and inserting to Y all the other nodes.

As far as the time complexity is concerned, the computational efficiency of push-relabel depends on how the active nodes are selected. In the case of any selection order, the algorithm computes a maximum flow in $O(n^2 m)$ time. Specific selection orders lead to algorithms running in $O(n^3)$, in $O(n^2\sqrt{m})$, and in $O(nmlog(n^2/m)$ time (the latter version is based on the dynamic tree data structure of Sleator and Tarjan (1983)). Additional details about the

algorithm and the related proofs can be found in Goldberg and Tarjan (1988) and in Ahuja, Magnanti, and Orlin (1993).

1.2 Extensions to parametric networks

In a *parametric network*, the arc capacities are functions of a real-valued parameter λ. Hereafter we will denote the parametric arc capacity function by c_λ.

Gallo, Grigoriadis, and Tarjan (1989) made the following assumptions:

- $c_\lambda(s, v)$ is a nondecreasing function of λ for each arc (s, v) (assuming $v \neq t$);
- $c_\lambda(v, t)$ is a nonincreasing function of λ for each arc (v, t) (assuming $v \neq s$);
- $c_\lambda(v, w)$ is constant otherwise.

The authors addressed the problem of computing the maximum flows (and the minimum cuts) for each value of an increasing sequence of parameter values $\lambda_1 < \lambda_2 < \cdots < \lambda_h$, where h is $O(n)$, when successive values are given on-line. At this purpose, they extended the push-relabel algorithm as follows. Suppose that, for some value λ_i, we have computed a maximum flow f and a valid distance d using push-relabel. When the value of the parameter increases to λ_{i+1}, the capacity of the arcs (s, v) may increase, whereas the capacity of the arcs (v, t) may decrease. Let us modify the current maximum flow f by replacing $f(v, t)$ by $min\{c_{\lambda_{i+1}}(v, t), f(v, t)\}$ for each $(v, t) \in E$, and replacing $f(s, v)$ by $max\{c_{\lambda_{i+1}}(s, v), f(s, v)\}$ for each $(s, v) \in E$ such that $d(v) < n$.

The modified f is a preflow. Furthermore, it is easy to verify that d is a valid distance for the modified f. Therefore, it is possible to compute a maximum flow (and a minimum cut) for λ_{i+1} applying the push-relabel algorithm starting with the modified f and the current d.

Since the distance labels never decrease during the running of the algorithm, then the total running time required by the h maximum flow computations is comparable to the time needed to solve a single maximum flow problem via the push-relabel algorithm (Gallo, Grigoriadis, and Tarjan, 1989). Therefore, depending on the order in which the active nodes are examined, the parametric maximum flow problem can be solved in $O(n^2m)$, $O(n^3)$, $O(n^2\sqrt{m})$ or $O(nmlog(n^2/m)$ time (if one uses the dynamic tree implementation).

Furthermore, since the distance labels never decrease, the minimum cuts produced via the procedure described in Section 1.1 satisfy the so-called "nesting" property, i.e., if (X_1, Y_1), (X_2, Y_2) $\ldots$, (X_h, Y_h) denote the minimum cuts computed by the algorithm, then $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_h$. This implies that, when the capacities are affine linear functions of the parameter, the parametric maximum flow value function, defining the maximum flow value (and, equivalently, the minimum cut capacity) as a function of the parameter, has at most $n - 1$ breakpoints. All the breakpoints can therefore be determined by $O(1)$ maximum flow computations (Gallo, Grigoriadis, and Tarjan, 1989).

Gusfield and Martel (1992) enhanced the results of Gallo et al. showing that, on the same kind of parametric networks, the above described algorithmic ideas can be extended to work, in the same time bound, even when the h values of the parameter are not presented in sorted order, and the corresponding maximum flows have to be computed on-line.

McCormick (1999) extended the results of Gallo, Grigoriadis, and Tarjan (1989) to parametric networks in which the subset of arcs with parametric capacity has a more general structure. The studied parametric networks come from some scheduling applications.

Consider a network with a distinguished out-arborescence S (i.e., a tree directed away from its root) rooted at the source s, and with an in-arborescence T (i.e., a tree directed towards its root) rooted at the sink t, where T is assumed to be arc-disjoint from S. Assume that, if $w \in S$, $w \neq s$, then the only arc entering w is the one in S. Similarly, if $v \in T$, $v \neq t$, then the only

arc outgoing from v is the one in T. Assume to have affine linear capacities $c_\lambda(v, w) = p_{(v,w)} + a_{(v,w)}\lambda$, such that $c_\lambda(v, w)$ is a nondecreasing function of λ for all $(v, w) \in S$, $c_\lambda(v, w)$ is a nonincreasing function of λ for all $(v, w) \in T$, and $c_\lambda(v, w)$ is a constant for all the other arcs (in his work, McCormick provided an equivalent description, where $c_\lambda(v, w)$ is a nonincreasing function of λ for all $(v, w) \in S$, and $c_\lambda(v, w)$ is a nondecreasing function of λ for all $(v, w) \in T$; we have chosen the reversed presentation for the sake of uniformity with the other approaches from the literature).

Finally, impose the following sort of superadditivity to the coefficients $a_{(v,w)}$:

- $a_{(v,w)} \geq \sum_{(w,i)\in S} a_{(w,i)}$, for each $(v, w) \in S$;
- $a_{(v,w)} \leq \sum_{(i,v)\in T} a_{(i,v)}$, for each $(v, w) \in T$.

It is easy to observe that, when S is the set of arcs outgoing from s and T is the set of arcs entering t, then the parametric networks studied by McCormick (1999) reduce to the ones studied by Gallo, Grigoriadis, and Tarjan (1989).

Now, consider a value λ_i of the parameter, and solve the maximum flow problem using the push-relabel algorithm previously reviewed. Let f be the returned maximum flow, and d be the corresponding valid distance. McCormick proved that, when the value of the parameter increases to the value λ_{i+1}, it is possible to modify the current maximum flow f in such a way to obtain a preflow. Also in this case, d remains a valid distance for the modified f. Therefore, also in this case it is possible to compute a maximum flow (and a minimum cut) for λ_{i+1} applying the push-relabel algorithm starting with the modified f and the current d, and the analysis performed by Gallo, Grigoriadis, and Tarjan (1989) works again.

In particular, the problem of computing the maximum flows (and the minimum cuts) for each value of an increasing sequence of parameter values $\lambda_1 < \lambda_2 < \cdots < \lambda_h$, where h is $O(n)$ and the successive values are given on-line, can be solved in the same time bound needed to solve a single maximum flow problem via the push-relabel algorithm.

Moreover, since the distance labels never decrease, the minimum cuts produced by the algorithm satisfy the nesting property and, again, the parametric maximum flow value function has at most $n - 1$ breakpoints.

Arai, Ueno, and Kajitani (1993) considered parametric networks where the capacities of the arcs incident to a node u^* (other than s and t) are nondecreasing functions of a parameter λ, whereas the other arc capacities are constant. The authors proved that, for this family of parametric networks, the minimum cuts satisfy a generalized nesting property, which can be stated as follows. Given a sequence of parameter values $\lambda_1 < \lambda_2 < \cdots < \lambda_h$, then there exist corresponding minimum cuts $(X_1, Y_1), (X_2, Y_2) \ldots, (X_h, Y_h)$ such that $X_{\sigma(1)} \subseteq X_{\sigma(2)} \subseteq \cdots \subseteq X_{\sigma(h)}$, for some permutation σ on $\{1, 2, \ldots, h\}$. In other words, a "totally ordered selection" of minimum cuts does exist (Brumelle, Granot, and Liu, 2002).

The first consequence is that also Arai et al.'s parametric networks are characterized by $O(n)$ different minimum cut capacities (and, so, $O(n)$ different maximum flow values). Therefore, when the arc capacities are affine linear functions of the parameter, the number of breakpoints of the parametric maximum flow value function is $O(n)$.

Furthermore, a suitable use of the push-relabel algorithm allows one to compute all the minimum cut capacities (and, so, all the maximum flow values) by $O(1)$ maximum flow computations. Such a property will be better described in Section 2.1. At this regard, however, we want to make in evidence that the authors did not discuss how to compute the maximum flows themselves.

2 A new extension

Here we will consider parametric networks such that:

- $c_\lambda(s, v)$ is a nondecreasing function of λ for each arc (s, v);
- $c_\lambda(v, t)$ is a nonincreasing function of λ for each arc (v, t);
- $c_\lambda(u^*, v)$ is a nondecreasing function of λ for each arc $(u^*, v) \in E$ leaving a *critical node* $u^* \neq s, t$;
- $c_\lambda(v, u^*)$ is a nonincreasing function of λ for each arc $(v, u^*) \in E$ entering the critical node u^*;
- $c_\lambda(v, w)$ is constant otherwise.

We will show that the minimum cuts of these parametric networks satisfy a relaxed form of the generalized nesting property introduced by Arai, Ueno, and Kajitani (1993). Using this property we will prove that, when the arc capacities are affine linear functions, the parametric maximum flow value function has at most $n - 1$ breakpoints. By the same arguments of Arai, Ueno, and Kajitani (1993) it is therefore possible to determine all the minimum cut capacities (and so, all the maximum flow values) by $O(1)$ maximum flow computations.

Then, we will enhance the results of Arai, Ueno, and Kajitani (1993) showing that, given any sequence of $O(n)$ increasing values of the parameter, it is possible to compute the corresponding maximum flows themselves in $O(1)$ maximum flow computations, by suitably extending Goldberg and Tarjan's maximum flow algorithm.

2.1 Finding all breakpoints

The purpose of this section is to show that the generalized nesting property of minimum cuts, proved by Arai, Ueno, and Kajitani (1993) for the special kind of parametric networks where the set of arcs whose capacity depends on the parameter is the set of arcs incident to a single node u^* (the capacities are nondecreasing functions of the parameter), holds true for the parametric networks considered in this work, although in a relaxed form. More precisely, the property states that, given any sequence of parameter values, there exists a set of corresponding minimum cuts which can be dichotomized in such a way that, for each set of the partition, the generalized nesting property holds true. In other words, for the studied networks, two "totally ordered selections" of minimum cuts do exist (Brumelle, Granot, and Liu, 2002). The proof is similar to the one devised by Arai, Ueno, and Kajitani (1993), and it bases on the nesting property of minimum cuts proved by Gallo, Grigoriadis, and Tarjan (1989). The nesting property will be therefore reviewed below.

Theorem 2.1 (Nesting property). *Consider the parametric networks studied by Gallo, Grigoriadis, and Tarjan* (1989)*, where the arcs leaving the source have a nondecreasing capacity function, the arcs entering the destination have a nonincreasing capacity function, whereas the other arcs have a constant capacity. For those networks, given a sequence of parameter values* $\lambda_1 < \lambda_2 < \cdots < \lambda_h$*, there exist corresponding minimum cuts* (X_1, Y_1)*,* $(X_2, Y_2) \ldots, (X_h, Y_h)$ *such that* $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_h$.

Theorem 2.2 (Relaxed generalized nesting property). *Given a sequence of parameter values* $\lambda_1, \lambda_2, \ldots \lambda_h$*, in the parametric networks studied in this work there exist corresponding minimum cuts* (X_1, Y_1)*,* $(X_2, Y_2) \ldots, (X_h, Y_h)$ *such that* $X_{\sigma(1)} \subseteq X_{\sigma(2)} \subseteq \cdots \subseteq X_{\sigma(q)}$ *and* $X_{\sigma(q+1)} \subseteq X_{\sigma(q+2)} \subseteq \cdots \subseteq X_{\sigma(h)}$*, for some permutation* σ *on* $\{1, 2, \ldots, h\}$.

Proof: Without loss of generality, assume that the critical node u^* belongs to the cutset Y_i for $i = 1, \ldots, q$, $\lambda_1 < \lambda_2 < \cdots < \lambda_q$, whereas u^* belongs to the cutset X_j, for $j = q+1, \ldots, h$, $\lambda_{q+1} < \lambda_{q+2} < \cdots < \lambda_h$.

Let G_{tu^*} be the network obtained from the original network G by identifying the nodes t and u^* as the new destination node. That is, G_{tu^*} is the network formed by shrinking t and u^* into a single node, eliminating loops and combining multiple arcs by adding their capacities (see also (Gallo, Grigoriadis, and Tarjan, 1989)). Similarly, let G_{su^*} be the network obtained from G by identifying the nodes s and u^* as the new source node.

In G_{tu^*}, the arcs leaving the source have a nondecreasing capacity function, whereas the ones entering the destination (the one obtained shrinking t and u^*) have a nonincreasing capacity function. Therefore, from Theorem 2.1, in G_{tu^*} there exist minimum cuts $(X_i, V \setminus X_i \setminus \{u^*\})$ relative to the parameter values $\lambda_i, i = 1, 2, \ldots, q$, such that $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_q$.

Similarly, always from Theorem 2.1, in G_{su^*} there exist minimum cuts $(Z_i, V \setminus Z_i \setminus \{u^*\})$ relative to the parameter values $\lambda_i, i = q+1, q+2, \ldots, h$, such that $Z_{q+1} \subseteq Z_{q+2} \subseteq \cdots \subseteq Z_h$. Let $X_i = Z_i \cup \{u^*\}$, $i = q+1, q+2, \ldots, h$. Clearly it is $X_{q+1} \subseteq X_{q+2} \subseteq \cdots \subseteq X_h$.

It is easy to show that $(X_i, V \setminus X_i)$ are minimum cuts in the original network G (obtained from G_{tu^*} by distinguishing node u^* from node t) for $\lambda_i, i = 1, 2, \ldots, q$. Similarly, $(X_i, V \setminus X_i)$ are minimum cuts in G for $\lambda_i, i = q+1, q+2, \ldots, h$. The thesis follows. □

If we assume the affine linearity of the arc capacities, from the above result we get:

Corollary 2.1. *The parametric maximum flow value function corresponding to the networks studied in this work has at most $n-1$ breakpoints.*

Proof: For each value of the parameter, the maximum flow value is given either by the parametric maximum flow value function corresponding to G_{tu^*}, or by the parametric maximum flow value function corresponding to G_{su^*} (as follows from the proof of Theorem 2.2). Therefore, the parametric maximum flow value function for the parametric network G is given by the minimum of these two parametric maximum flow value functions. Since both functions have $O(n)$ breakpoints, such a minimum function also has $O(n)$ breakpoints. □

Furthermore:

Corollary 2.2. *The $O(n)$ breakpoints can be computed by $O(1)$ maximum flow computations.*

Proof: Let us determine the $O(n)$ breakpoints of the parametric maximum flow value functions corresponding to the parametric networks G_{tu^*} and G_{su^*}, respectively, using the parametric maximum flow algorithm as suggested by Gallo, Grigoriadis, and Tarjan (1989). This can be obtained by performing $O(1)$ maximum flow computations.

Since the parametric maximum flow value function for the original network G, and its breakpoints, can be obtained by computing the minimum of the two functions corresponding to G_{tu^*} and G_{su^*}, the thesis follows. □

2.2 The new parametric maximum flow algorithm

An interesting question is how we can suitably extend the push-relabel algorithm in order to solve the parametric maximum flow problem on the networks introduced in Section 2.

As in the previous situations, consider the problem of computing the maximum flows (and the corresponding minimum cuts) for each value of an increasing sequence of parameter

values $\lambda_1 < \lambda_2 \cdots < \lambda_h$, where h is $O(n)$. Successive values can be given on-line. At this purpose, let us extend the push-relabel algorithm as follows.

At the first step, compute a maximum flow f and a valid distance d for the value of the parameter λ_1, using the classical push-relabel algorithm as proposed by Goldberg and Tarjan (1988). Let k be the distance label of the critical node u^* ($d(u^*) = k$).

When the value of the parameter increases to λ_2, the capacity of the arcs of type (s, v) and (u^*, v) may increase, whereas the capacity of the arcs of type (v, t) and (v, u^*) may decrease. As a consequence, the flow modifications suggested by Gallo et al. may not work. Assume, in fact, to modify f by replacing $f(v, t)$ by $min\{c_{\lambda_2}(v, t), f(v, t)\}$ for each $(v, t) \in E$, and replacing $f(s, v)$ by $max\{c_{\lambda_2}(s, v), f(s, v)\}$ for each $(s, v) \in E$.

Furthermore, assume to extend the above modifications to the arcs incident node u^*, by replacing $f(v, u^*)$ by $min\{c_{\lambda_2}(v, u^*), f(v, u^*)\}$ for each $(v, u^*) \in E$, and replacing $f(u^*, v)$ by $max\{c_{\lambda_2}(u^*, v), f(u^*, v)\}$ for each $(u^*, v) \in E$.

With respect to the previous analysed parametric networks, the modified f is not necessarily a preflow. In fact, after the flow transformation, either u^* remains a balanced node, and in such a case we still have a preflow, or u^* can become a destination node, in the sense that the total flow outgoing from u^* can be strictly greater than the total entering flow.

Furthermore, the current d is not necessarily a valid distance for the modified f, due to the following types of residual arcs:

- residual arcs (v, u^*) corresponding to arcs $(u^*, v) \in E$; this can happen only if the flow on (u^*, v) was zero before increasing the parameter value to λ_2;
- residual arcs (v, s) corresponding to arcs $(s, v) \in E$; this can happen only if the flow on (s, v) was zero before increasing the parameter value to λ_2 (observe that we modified the flow of each arc (s, v) independently of the label of node v).

In order to overcome these difficulties, which prevent the application of the "pure" push-relabel algorithm starting with the modified f and the current d, we propose to perform the following step, consisting in suitably modifying d:

(*initialization step*) if $d(u^*) = k < n$, set $d(u^*) = n$; in any case, increase the distance label of the source node s, $d(s) = n$, to the value $n + d(u^*)$.

In our approach, such a step is performed once, that is the first time, after the execution of the flow modifications, the total flow outgoing from u^* is strictly greater than the total entering flow (before that, we can apply the parametric maximum flow algorithm devised by Gallo et al.). W.l.o.g., we will assume here that the above described condition concerning node u^* verifies at the second maximum flow computation, i.e., in correspondence of the value λ_2 of the parameter.

After the initialization step, the proposed parametric algorithm performs the following flow modifications:

(*flow modification step*):

- replace $f(v, t)$ by $min\{c_{\lambda_2}(v, t), f(v, t)\}$ for each $(v, t) \in E$, and replace $f(s, v)$ by $max\{c_{\lambda_2}(s, v), f(s, v)\}$ for each $(s, v) \in E$ such that $d(v) < d(s)$;
- replace $f(v, u^*)$ by $min\{c_{\lambda_2}(v, u^*), f(v, u^*)\}$ for each $(v, u^*) \in E$, and replace $f(u^*, v)$ by $max\{c_{\lambda_2}(u^*, v), f(u^*, v)\}$ for each $(u^*, v) \in E$ such that $d(v) < n$,

and applies the push-relabel algorithm by managing the critical node u^* as a destination node; in other words, u^* will be never selected by the push-relabel algorithm.

The first observation is that, since u^* is handled as a destination node, then the modified f can be interpreted as a preflow (with respect to the two destinations t and u^*). Furthermore, thanks to the initialization step, d is a valid distance function for the modified f. This is true for each residual arc, with the only (possible) exception of the residual arcs (u^*, v) corresponding to arcs (v, u^*) of G: in fact, since $d(u^*)$ is augmented to n when the original value k was $< n$, then for, those arcs, it may be $d(u^*) > d(v) + 1$. Observe however that, since u^* will be handled as a destination node, the residual arcs leaving u^* will be never selected by the push-relabel algorithm, and so they can be (temporarily) removed from the residual graph.

Due to the above considerations, we can properly apply the push-relabel algorithm starting with the modified f and d.

Informally speaking, we have transformed the problem of computing a maximum flow from s to t after the modification of the arc capacities (due to the setting $\lambda = \lambda_2$) into the problem of sending the maximum amount of flow, from s and from the nodes v adjacent to u^* (i.e., the nodes v such that $(u^*, v) \in E$), towards t and towards u^*. We will prove that, thanks to the setting of $d(s)$ made by the initialization step, the excesses of flow in the network are sent towards t and towards u^* by giving priority to the destination t; in other words, the excesses of flow are sent towards u^* only if no residual path exists from the selected active node to t. Moreover, the flow excess is rerouted towards s only if no residual path exists either towards t or towards u^*. As a consequence, the proposed variation of the push-relabel algorithm correctly sends the maximum amount of flow from s and from the nodes adjacent to u^* towards t (with priority) and towards u^*.

When the push-relabel algorithm terminates, the resulting f is not necessarily the maximum flow corresponding to the value λ_2 of the parameter, due to node u^*. In fact, by the way in which this special node is handled, it may happen that the flow outgoing from u^* exceeds the one pushed towards u^*, or viceversa. We will refer to it as a maximum pseudoflow. In both cases, it is necessary to balance node u^*, by reducing the flow sent to t, in the first case, and sending back to s the surplus of flow, in the second case (by looking at the original network G). So doing, the resulting f will be converted into a maximum flow relative to $\lambda = \lambda_2$.

The proposed approach therefore considers separately the phase in which the maximum pseudoflow is computed, from the one in which the maximum pseudoflow is converted into a maximum flow. The major motivation is that the parametric maximum flow algorithm begins the next iteration, devoted to the computation of the maximum flow corresponding to the value λ_3 of the parameter, starting with the maximum pseudoflow f (i.e., the *flow modification step* is applied to such a maximum pseudoflow), and with the corresponding valid distance d, found for $\lambda = \lambda_2$. In fact, the transformation of the maximum pseudoflow into a maximum flow, via the sending of the excess of flow at u^* back to s, or the reduction of flow from u^* to t, migth invalidate the current valid distance d, and so prevent its use for the next maximum flow computation. On the other hand, when the maximum pseudoflow is modified by the *flow modification step* due to the setting $\lambda = \lambda_3$, then d remains valid for f. Observe that, after the flow modification step for $\lambda = \lambda_3$, the critical node u^* could become active, balanced, or it could have a deficit of flow. In our approach, we will treat u^* as a destination in every situation (that is, we will push the maximum amount of flow from u^*, and compute the maximum amount of flow towards u^*, which can not be pushed directly to t), in order to be able to use, at each step, the previously computed valid distance d, so amortizing the computing time spent over all the values of the parameter.

In conclusion, after the maximum flow computation for $\lambda = \lambda_1$ and after the initialization step, the two phases are iterated for each value of the sequence $\lambda_2 < \cdots < \lambda_h$. More precisely, at each iteration, the first phase is applied to the maximum pseudoflow f, and

to the corresponding valid distance d, computed for the previous value of the parameter, by suitably modifying f as indicated by the *flow modification step* (in the case $\lambda = \lambda_2$, such a maximum pseudoflow is the maximum flow found for $\lambda = \lambda_1$). Then, the second phase converts the computed maximum pseudoflow into a maximum flow. The parametric maximum flow algorithm is sketched below:

Parametric maximum flow algorithm

- compute a maximum flow f and a valid distance d for $\lambda = \lambda_1$ applying the classical push-relabel algorithm;
- perform the initialization step;
- for $\lambda = 2, \ldots, h$:

1. (*first phase*) modify f by performing the flow modification step and apply the variant of the push-relabel algorithm (where u^* is managed as a destination node), returning a maximum pseudoflow f and a corresponding valid distance d;
2. (*second phase*) convert f into a maximum flow.

In the following section we will prove the correctness of the two phases. Then, we will suggest some implementation issues, and derive the time complexity of the whole algorithmic approach.

2.3 The algorithm correctness

Given a value of the parameter, consider the first phase of the proposed approach, consisting of applying the push-relabel algorithm starting with the modification of the current maximum pseudoflow flow f (via the *flow modification step*) and the corresponding valid distance d. Remember that the first phase handles the critical node u^* as a destination node.

The following properties hold true.

Property 2.1. *The algorithm sends flow towards t and towards u^* by giving priority to the destination t; in other words, the excesses of flow are sent towards u^* only if no residual path exists from the selected active node to t.*

Proof: Consider an active node v. Since d is a valid distance function, then $d(v)$ estimates, from below, the length of the shortest augmenting paths from v to t. Therefore, if $d(v) \geq n$, then no residual path exists from v to t. Now, the algorithm can push flow from v to u^*, along a residual path, only if $d(v) \geq d(u^*)$. Due to the setting of $d(u^*)$, this can happen only if $d(v) \geq n$, that is, no residual path exists from v to t. □

Property 2.2. *The algorithm reroutes flow towards u^* only if it is not possible to push further flow from u^* to t. Moreover, the algorithm reroutes flow towards s only if it is not possible to push further flow from s towards t or towards u^*.*

Proof: At the beginning, the algorithm saturates each arc (u^*, v) such that $d(v) < n$ (since, otherwise, no residual path exists from v to t). During the algorithm running, flow can be rerouted towards u^*, along a residual arc (v, u^*), only if $d(v) = d(u^*) + 1 > n$. Therefore, this happens only if no residual path exists from v to t.

Similarly, at the beginning the algorithm saturates each arc (s, v) such that $d(v) < d(s) = d(u^*) + n$ (since, otherwise, no residual path exists towards the two destinations: neither from v to t, nor from v to u^*). During the algorithm running, flow can be rerouted towards s,

along a residual arc (v, s), only if $d(v) = d(s) + 1 = d(u^*) + n + 1$. Therefore, this happens only if no residual path exists from v either to t or to u^*. □

The above properties can be used to state the following result:

Theorem 2.3. *The proposed variation of the push-relabel algorithm correctly sends the maximum amount of flow from the active nodes adjacent to s, and from the active nodes adjacent to u^*, towards t. Moreover, the algorithm sends to u^* the maximum amount of flow which can not be sent directly to t.*

Proof: It follows from Property 2.1 and from Property 2.2. □

Therefore:

Corollary 2.3. *After the balancing of node u^*, performed during the second phase, the resulting flow f is a maximum flow from s to t for the given value of the parameter.*

Proof: From Theorem 2.3, the first phase determines a cut (X, Y) such that both s and u^* belong to X, node t belongs to Y, and such that all the arcs from X to Y are saturated, whereas the arcs from Y to X have a zero flow. If node u^* is balanced during the second phase by sending the additional entering flow back to s, then the flow crossing (X, Y) does not change: (X, Y) is therefore a minimum cut, and the returned flow is a maximum flow. Consider now the case in which node u^* has an excess of outgoing flow, and therefore it is balanced during the second phase by reducing flow from u^* to t. Always from Theorem 2.3, the first phase determines an alternative cut, say (Z, W), such that s belongs to Z, u^* and t belong to W, and such that all the arcs from Z to W are saturated, whereas the arcs from W to Z have a zero flow. If u^* is balanced by reducing flow from u^* to t, then the flow crossing (Z, W) does not change: (Z, W) is therefore an alternative minimum cut, and the returned flow is a maximum flow. The proof follows. □

2.4 The computational time complexity

For each node v, $d(v)$ never decreases during the running of the first phases. Therefore, the $O(n)$ first phases totally require $O(1)$ maximum flow computations. Specifically, as for the parametric networks previously analyzed, these phases can be performed in $O(n^2m)$, $O(n^3)$, $O(n^2\sqrt{m})$ or $O(nmlog(n^2/m)$ time, depending on the order in which the active nodes are selected in order to perform the push-relabel operations.

As far as the second phases are concerned, each phase of this type can be implemented in $O(nm)$ time (Goldberg and Tarjan, 1986). This can be achieved by performing push-relabel operations which are simpler than the ones previously described, because the arc flows need only to be reduced and never increased. In this way, the overall time complexity of the proposed parametric maximum flow algorithm is $O(n^2m)$.

However, as observed in Goldberg and Tarjan (1986), it is possible to refine this result as follows. At the end of each first phase, firstly eliminate the cycles of flows from the returned pseudoflow, so converting the subgraph of G induced by the arcs with a positive flow to an acyclic graph, say G^*. This can be performed in $O(mlogn)$ time by means of an algorithm proposed by Sleator and Tarjan (1985). Then, convert the pseudoflow into a flow by processing the nodes of G^* in a reverse topological order, in $O(m)$ time. Using this

more sophisticated implementation, the overall time complexity of the proposed parametric maximum flow algorithm becomes $O(nmlogn)$. Therefore:

Theorem 2.4. *The parametric maximum flow problem on the extended parametric networks can be solved in $O(nmlogn)$ time.*

A final observation follows. Consider the parametric networks introduced by Arai, Ueno, and Kajitani (1993). Let us modify the initialization step of the proposed parametric algorithm as follows:

(*modified initialization step*) set $d(u^*)$ to $max\{k, n, l-1\}$, where $l = max\{d(v) : (v, u^*) \in E\}$; increase the distance label of the source node s, $d(s) = n$, to the value $n + d(u^*)$,

and perform the following simplification of the flow modification step:

(*simplified flow modification step*):

- replace $f(u^*, v)$ by $max\{c_{\lambda_2}(u^*, v), f(u^*, v)\}$ for each $(u^*, v) \in E$ such that $d(v) < n$.

It is easy to observe that the resulting variation of the parametric algorithm is able to solve the parametric maximum flow problem, with the same time complexity, when applied to the parametric networks of Arai et al. Theorem 2.4 thus applies also to those kinds of networks.

3 Conclusion

The paper addressed a parametric maximum flow problem where the parametric arc capacities are associated with the arcs leaving the source and entering the sink, as in Gallo, Grigoriadis, and Tarjan (1989), and with the arcs incident a critical node. For such a node, the entering arcs have a nonincreasing capacity, whereas the leaving arcs have a nondecreasing capacity. We showed that, on this type of parametric networks, the parametric maximum flow problem can be solved via $O(1)$ maximum flow computations, by suitably extending the preflow-push method of Goldberg and Tarjan. A variation of the proposed algorithm is able to solve the parametric maximum flow problem, via $O(1)$ maximum flow computations, also for the parametric networks of Arai, Ueno, and Kajitani (1993).

Observe that the proposed parametric algorithm could be useful within a reoptimization context, when some arc capacities are modified for some arcs incident the source, the sink, or a certain critical node, and the maximum flow has to be computed for $O(n)$ changes of the arc capacities. Provided that the arc capacities vary along the scheme addressed in this work, the parametric maximum flow algorithm allows one to solve this kind of maximum flow reoptimization in $O(nmlogn)$ time. Note that the same result holds true in the case in which the values of the parameter are given in a decreasing order, and the parametric network is reversed with respect to the described representation; in other words, the parametric network is such that the capacities of the arcs leaving the source are nonincreasing, the capacities of the arcs entering the sink are nondecreasing, and the critical node is such that the capacities of the entering arcs are nondecreasing, whereas the ones of the leaving arcs are nonincreasing (to obtain that, reverse the direction of the arcs, and compute a maximum flow from t to s, as suggested in Gallo, Grigoriadis, and Tarjan (1989)).

In order to conclude this note, observe that it is not immediate to generalize the obtained results neither to parametric networks in which the capacities of the arcs incident the critical node vary in a different way (for example, they are all decreasing), nor to parametric networks

in which there are more critical nodes. The main motivation is that, in those cases, flow modifications like the ones suggested in this paper may create more "destination" nodes: such nodes can not be managed independently, but they have to co-operate in sending the flow. We plan to analyze these more general parametric networks, in order to look for special cases for which the parametric maximum flow problem (as well as the related arc capacity reoptimization problem) can be efficiently solved. As suggested by an anonymous referee, another interesting avenue of research might be to analyse the all pairs maximum flow values under some arc capacity variations.

Acknowledgments We thank Prof. D. Granot and Prof. S.T. McCormick for their comments on a previous version of this note.

References

Ahuja, R.K., T.L. Magnanti, and J.B. Orlin. (1993). *Network Flows: Theory, Algorithms and Applications*. Englewood Cliffs, NJ: Prentice Hall.

Arai, T., S. Ueno, and Y. Kajitani. (1993). "Generalization of a Theorem on the Parametric Maximum Flow Problem." *Discrete Applied Mathematics*, 41, 69–74.

Brumelle, S., D. Granot, and L. Liu. (2002). "Ordered Optimal Solutions and Parametric Minimum Cut Problems." Working paper.

Carstensen, P.J. (1983). "Complexity of Some Parametric Integer and Network Programming Problems." *Mathematical Programming*, 26, 64–75.

Ford Jr., L.R. and D.R. Fulkerson. (1962). *Flows in Networks*. Princeton, NJ: Princeton University Press.

Gallo, G., M.D. Grigoriadis, and R.E. Tarjan. (1989). "A Fast Parametric Maximum Flow Algorithm and Applications." *SIAM J. Comp.*, 18(1), 30–55.

Goldberg, A.V. and R.E. Tarjan. (1986). "A New Approach to the Maximum Flow Problem." *Proc. 18th Annual ACM Symposium on Theory of Computing*, 136–146.

Goldberg, A.V. and R.E. Tarjan. (1988). "A New Approach to the Maximum Flow Problem." *JACM*, 35(4), 921–940.

Gusfield, D. and C. Martel. (1992). "A Fast Algorithm for the Generalized Parametric Minimum Cut Problem and Applications." *Algorithmica*, 7, 499–519.

McCormick, S.T. (1999). "Fast Algorithms for Parametric Scheduling Come From Extensions to Parametric Maximum Flow." *Operations Research*, 47(5), 744–756.

Sleator, D.D. and R.E. Tarjan. (1983). "A Data Structure for Dynamic Trees." *J. Comput. System Sci.*, 24, 362–391.

Sleator, D.D. and R.E. Tarjan. (1985). "Self-Adjusting Binary Search Trees." *JACM*, 32, 652–686.