Applied Mathematics, 2013, 4, 3-10
http://dx.doi.org/10.4236/am.2013.410A1002 Published Online October 2013 (http://www.scirp.org/journal/am)

Partitioning Algorithm for the Parametric Maximum Flow

Mircea Parpalea[1], Eleonor Ciurea[2]
[1]National College Andrei Şaguna, Braşov, Romania
[2]Department of Theoretical Computer Science, Transilvania University of Braşov, Braşov, Romania
Email: parpalea@gmail.com, e.ciurea@ unitbv.ro

Received May 27, 2013; revised June 27, 2013; accepted July 4, 2013

ABSTRACT

The article presents an approach to the maximum flow problem in parametric networks with linear capacity functions of a single parameter, based on the concept of shortest conditional augmenting directed path. In order to avoid working with piecewise linear functions, our approach uses a series of parametric residual networks defined for successive subintervals of the parameter values where the parametric residual capacities of all arcs remain linear functions. Besides working with linear instead piecewise linear functions, another main advantage of our approach is that every directed path in such a parametric residual network is also a conditional augmenting directed path for the subinterval for which the parametric residual network was defined. The complexity of the partitioning algorithm is $O\left(Kn^2m\right)$ where K is the number of partitioning points of the parameter values interval, n and m being the number of nodes, respectively the number of arcs in the network.

Keywords: Network Flow; Parametric Flow; Conditional Augmenting Paths

1. Introduction

Efficient algorithms for computing maximum flows in networks are important not only because they are applied directly to the analysis of traffic or communication networks, but also because they are often employed as subproblems in other general network problems. Fundamental algorithms for network flow were designed and efficient algorithms exist (Ahuja, Magnanti, & Orlin) [1] to solve different instances of this problem. A natural generalization of the maximum flow problem can be obtained by making the capacities of some arcs functions of a single parameter. The parametric maximum flow problem is to compute all maximum flows for every possible value of the parameter. For the parametric maximum flow problem with zero lower bounds and linear capacity functions of a single parameter, Hamacher and Foulds [2] investigated an approach for determining in each iteration an improvement of the flow defined on the whole interval of the parameter while for the same problem, Ruhe [3], [4] proposed a "piece-by-piece" approach. The partitioning type approach, which is presented in this paper, proposes an original algorithm for computing the maximum flow in networks with constant lower bounds and linear upper bound functions.

Partitioning technique in network has been, in the latest years, a more and more active research topic in both engineering and theoretical research. The reason why the problem under consideration is of genuine practical and theoretical interest lies in that graph partitioning applications are described on a wide variety of subjects as: data distribution in parallel-computing, VLSI circuit design, image processing, computer vision, route planning, air traffic control, mobile networks, social networks, etc. [5]. Unfortunately, graph partitioning is an NP-hard problem, and therefore all known algorithms for generating partitions merely return approximations to the optimal solution.

Further on, this paper is organized as follows; Section 2 presents the basic network flow terminology and results used in the rest of the paper. More specialized terminology is developed in later sections. In Section 3, we introduce the parametric maximum flow problem and Section 4 presents the partitioning algorithm for solving this problem. Finally, Section 5 gives an example of how the algorithm works on a network with linear upper bound functions of a single parameter. In the presentation to follow, some familiarities with flow algorithms are assumed and many details are omitted, since they are straight forward modifications of known results. Further details on

notions and results presented in Section 2 can be found in the papers of Ahuja *et al*. [1] and Ciurea *et al*. [6,7].

2. Terminology and Preliminaries

Let $G=(N,A,\ell,u,s,t)$ be a capacitated network with $n=|N|$ nodes and $m=|A|$ arcs, $N=\{\cdots,i,\cdots\}$ being the set of nodes i and $A=\{\cdots,a,\cdots\}$ being the set of arcs a, so that for every arc in A, $a=(i,j)$ with $i,j\in N$. The *upper bound* function and the *lower bound* function are two nonnegative functions, $u(a)$ and $\ell(a)$ associated with each arc $a\in A$. The network has two special nodes: a source node s and a sink node t. *A* flow is a function $f:A\to\Re^{+}$ satisfying the next conditions:

$$\sum_{j|(i,j)\in A} f(i,j)-\sum_{j|(j,i)\in A} f(j,i)=\begin{cases} v, & i=s \\ 0, & i\neq s,t \\ -v, & i=t \end{cases} \quad (1)$$

for some $v\geq 0$, where v is referred to as the value of the flow f. Any flow on a directed network satisfying the flow bound constraints:

$$\ell(i,j)\leq f(i,j)\leq u(i,j),\quad \forall(i,j)\in A \quad (2)$$

is referred to as a *feasible flow*. A *cut* is a partition of the node set N into two subsets S and $T=N-S$, denoted by $[S,T]$. An arc $(i,j)\in A$ with $i\in S$ and $j\in T$ is referred to as a *forward arc* of the cut while an arc $(i,j)\in A$ with $i\in T$ and $j\in S$ as a *backward arc* of the cut. Let (S,T) denote the set of forward arcs in the cut and (T,S) denote the set of backward arcs. A cut $[S,T]$ is an $s-t$ *cut* if $s\in S$ and $t\in T$. The maximum flow problem is to determine a flow $\tilde{f}$ for which v is maximized. The maximum flow problem in a network can be solved in two phases: (1) establishing a feasible flow; (2) from a given feasible flow, establishing the maximum flow. For the first phase, see the algorithms presented in [1,7,8].

3. The Parametric Maximum Flow

The parametric flow problem consists in generalising the classic problem of flows in networks by transforming the upper bounds of some arcs $(i,j)\in A$ of the network $G=(N,A,\ell,u,s,t)$ in linear functions of a real parameter λ.

Definition 1. *A directed network* $G=(N,A,\ell,u,s,t)$ *for which the upper bounds* u *of some arcs* $(i,j)\in A$ *are functions of a real parameter* λ *is referred to as a parametric network and is denoted by* $\overline{G}=(N,A,\ell,\overline{u},s,t)$.

For a parametric network $\overline{G}$, the *parametric upper bound* (*capacity*) function $\overline{u}:A\times[0,\Lambda]\to\Re^{+}$ associates to each arc $(i,j)\in A$ and for each of the parameter values λ in an interval $[0,\Lambda]$, the real number $\overline{u}(i,j;\lambda)$, referred to as the *upper bound* of arc (i,j):

$$\overline{u}(i,j;\lambda)=u_0(i,j)+\lambda\cdot U(i,j),\quad \lambda\in[0,\Lambda]. \quad (3)$$

where $U:A\to\Re$ is a real valued function associating to each arc $(i,j)\in A$ the real number $U(i,j)$, referred to as the *parametric part of the upper bound* of the arc (i,j). The nonnegative value $u_0(i,j)$ is the upper bound of the arc (i,j) for $\lambda=0$, *i.e.* $\overline{u}(i,j;0)=u_0(i,j)$ with $\ell(i,j)\leq u_0(i,j)$. For the problem to be correctly formulated, the upper bound function of every arc $(i,j)\in A$ must respect the condition $\ell(i,j)\leq\overline{u}(i,j;\lambda)$ for the entire interval of the parameter values, *i.e.* $\forall(i,j)\in A$ and $\forall\lambda\in[0,\Lambda]$. It follows that the parametric part of the upper bounds $U(i,j)$ must satisfy the constraint: $U(i,j)\geq(\ell(i,j)-u_0(i,j))/\Lambda$, $\forall(i,j)\in A$. The parametric flow value function $\overline{v}:N\times[0,\Lambda]\to\Re$ associates to each of the nodes $i\in N$ a real number $\overline{v}(i;\lambda)$ referred to as the value of node i for each of the parameter λ values.

Definition 2. *A feasible flow in the parametric network* $\overline{G}=(N,A,\ell,\overline{u},s,t)$ *is called a* parametric flow *and it is a function* $\overline{f}:A\times[0,\Lambda]\to\Re^{+}$ *satisfying the following constraints:*

$$\sum_{j|(i,j)\in A}\overline{f}(i,j;\lambda)-\sum_{j|(j,i)\in A}\overline{f}(j,i;\lambda)=\overline{v}(i;\lambda),\quad \forall i\in N,\forall\lambda\in[0,\Lambda], \quad (4)$$

$$\ell(i,j)\leq\overline{f}(i,j;\lambda)\leq\overline{u}(i,j;\lambda),\quad \forall(i,j)\in A,\ \forall\lambda\in[0,\Lambda] \quad (5)$$

where $\sum_{i\in N}\overline{v}(i;\lambda),\ \forall\lambda\in[0,\Lambda]$.

The parametric maximum flow (PMF) problem is to compute all maximum flows for every possible value of $\forall\lambda\in[0,\Lambda]$:

$$\text{maximize}\quad \overline{v}(\lambda)\quad \text{for all}\ \lambda\in[0,\Lambda], \quad (6)$$

$$\sum_{j|(i,j)\in A}\overline{f}(i,j;\lambda)-\sum_{j|(i,j)\in A}\overline{f}(j,i;\lambda)=\begin{cases}\overline{v}(\lambda), & i=s\\ 0, & i\neq s,t\\ -\overline{v}(\lambda), & i=t\end{cases} \quad (7)$$

$$\ell(i,j)\leq\overline{f}(i,j;\lambda)\leq\overline{u}(i,j;\lambda),\forall(i,j)\in A \quad (8)$$

This problem looks like a classic maximum flow problem with the decisive difference that the variables $\overline{f}(i,j;\lambda)$ of this problem are piecewise linear functions instead of real numbers and that the upper bounds $\overline{u}(i,j;\lambda)$ are linear functions instead of constants.

Definition 3. *Let* **F** *be the set of piecewise linear functions* f_i *with* $f_i:[0,\Lambda]\to\Re^+$. *On the set* **F**, *an ordering relation is defined as follows*:

$$f_i \le f_j \Leftrightarrow f_i(\lambda)\le f_j(\lambda),\ \forall\lambda\in[0,\Lambda]. \quad (9)$$

For any two piecewise linear functions f_1 and f_2, it is possible that neither the relation $f_1\le f_2$ nor $f_2\le f_1$ hold for the entire interval $[0,\Lambda]$ and consequently, the two functions may not necessarily be comparable. But it is always possible that a partitioning *B*: $B: 0=\lambda_0<\lambda_1<\cdots<\lambda_{K+1}=\Lambda$ of the interval $[0,\Lambda]$ to be defined such as on every subinterval $[\lambda_k,\lambda_{k+1}]$, $k=0,\cdots,K$ one of the two cases to hold: $f_1\le f_2$ or $f_2\le f_1$, *i.e.* the two linear functions to become comparable. This means that the two functions have no crossing points within any subinterval $(\lambda_k,\lambda_{k+1})$, the only crossing points taking place for $\lambda_k, k=0,\cdots,K$.

Proposition 1. *For the parametric maximum flow problem, the subintervals* $J_k=[\lambda_k,\lambda_{k+1}]$, $k=0,\cdots,K$, *of the parameter* λ *values can be defined so that a minimum* $s-t$ *cut in the non-parametric network* $G_k=(N,A,\ell,u_k,s,t)$, *with* $u_k(i,j)=\overline{u}(i,j;\lambda_k)$, *also to represent a minimum* $s-t$ *cut for all the parameter* λ *values within the subinterval* J_k.

Definition 4. *A parametric* $s-t$ *cut partitioning, denoted by* $[S_k;J_k]$, $k=0,\cdots,K$, *is defined as a finite set of cuts* $[S_k,T_k]$, $k=0,\cdots,K$, *together with a partitioning of the interval* $[0,\Lambda]$ *of the parameter in disjoints subintervals* J_k, $k=0,\cdots,K$, *so that* $J_0\cup\cdots\cup J_K=[0,\Lambda]$.

Definition 5. *For the parametric maximum flow problem, the capacity* $\tilde{\overline{c}}[S_k;J_k]$ *of a parametric* $s-t$ *cut partitioning is a linear function on every subinterval* J_k, $k=0,\cdots,K$, *defined as*:

$$\tilde{\overline{c}}[S_k;J_k]=\sum_{(i,j)\in(S_k,T_k)}\overline{u}(i,j;\lambda)-\sum_{(j,i)\in(T_k,S_k)}\ell(j,i),\quad k=0,\cdots,K \quad (10)$$

Definition 6. *A parametric* $s-t$ *cut partitioning* $[S_k;J_k]$ *with the subintervals* J_k *assuring that every cut is a minimum cut* $[\tilde{S}_k,\tilde{T}_k]$ *within the subinterval* $[\lambda_k,\lambda_{k+1}]$ *is referred to as a parametric minimum* $s-t$ *cut and is denoted by* $[\tilde{S}_k;J_k]$, $k=0,\cdots,K$.

Theorem 1. (Parametric max-flow min-cut theorem [9]) *If there is a feasible flow in the parametric network* $\overline{G}$, *the value function* $\tilde{\overline{v}}$ *of the parametric maximum flow* $\tilde{\overline{f}}$ *from a source s to a sink t equals the capacity* $\tilde{\overline{c}}$ *of the parametric minimum* $s-t$ *cut* $[\tilde{S}_k;J_k]$, $k=0,\cdots,K$.

Let $\overline{f}=(\cdots\overline{f}(i,j),\cdots)_{(i,j)\in A}$ be a vector of feasible flow functions. Assuming that an arc $(i,j)\in A$ carries a flow $\overline{f}(i,j;\lambda)$, the existing flow can be increased either by pushing the flow $\overline{u}(i,j;\lambda)-\overline{f}(i,j;\lambda)$ from node i to node j over the arc (i,j), or by pulling the flow $\overline{f}(j,i;\lambda)-\ell(j,i)$ from node i to node j along the arc (j,i). These flows are computed as differences between piecewise linear functions of λ.

Definition 7. *For the parametric maximum flow problem, the parametric residual capacity* $\tilde{\overline{r}}(i,j;\lambda)$ *of any of the arcs* $(i,j)\in A$ *with respect to a given parametric flow* $\overline{f}$ *represents the maximum additional flow that can be sent from node* i *to node* j *over the arcs* (i,j) *and* (j,i) *and is given by*:

$$\tilde{\overline{r}}(i,j;\lambda)=\overline{u}(i,j;\lambda)-\overline{f}(i,j;\lambda)+\overline{f}(j,i;\lambda)-\ell(j,i). \quad (11)$$

Definition 8. *The subintervals* $\tilde{I}(i,j)\subseteq[0,\Lambda]$ *where an augmentation of the flow* $\overline{f}(i,j;\lambda)$ *is possible along the arc* (i,j) *are defined as follows*:

$$\tilde{I}(i,j)=\{\lambda\,|\,\tilde{\overline{r}}(i,j;\lambda)>0\}\ \text{ for }(i,j)\in A. \quad (12)$$

Definition 9. *Given a feasible flow* $\overline{f}$ *in the parametric network* $\overline{G}$, *the network denoted by* $\tilde{\overline{G}}(\overline{f})=(N,\tilde{A}(\overline{f}))$, *with* $\tilde{A}(\overline{f})=\{(i,j)\,|\,(i,j)\in A,\tilde{I}(i,j)\ne\Phi\}$ *being the set consisting only of arcs with positive parametric residual capacities, is referred to as the parametric residual network with respect to the given flow* $\overline{f}$ *for the parametric maximum flow problem.*

If an arc $(i,j)\in A$ does not belong to $\tilde{\overline{G}}(\overline{f})$ then $\tilde{I}(i,j):=\Phi$ is set.

Definition 10. *A conditional augmenting directed path is denoted by* $\tilde{\overline{P}}$ *and is a directed path* $\tilde{P}$ *from the source s to the sink t in the parametric residual network* $\tilde{\overline{G}}(\overline{f})$ *with the restriction that:*

$$\tilde{I}(\tilde{\overline{P}})=\bigcap_{(i,j)\in\tilde{\overline{P}}}\tilde{I}(i,j)\ne\Phi. \quad (13)$$

Definition 11. *A partly conditional augmenting directed path is denoted by* $\tilde{\overline{P}}(i)$ *and is a conditional augmenting directed path* $\tilde{\overline{P}}$ *from the source s to node* $i\ne t$ *in the parametric residual network* $\tilde{\overline{G}}(\overline{f})$.

Definition 12. *The parametric residual capacity of a conditional augmenting directed path* $\tilde{\overline{P}}$ *is the inner envelope of the parametric residual capacity functions* $\tilde{\overline{r}}(i,j;\lambda)$ *for all arcs* (i,j) *composing* $\tilde{\overline{P}}$ *and for all parameter* λ *values in the subinterval* $\tilde{I}(\tilde{\overline{P}})$:

$$\tilde{\overline{r}}(\tilde{\overline{P}};\lambda)=\min_{\lambda\in\tilde{I}(\tilde{\overline{P}})}\{\tilde{\overline{r}}(i,j;\lambda)\,|\,(i,j)\in\tilde{\overline{P}}\}. \quad (14)$$

Let $K\left(\tilde{\bar{P}}\right)$ be the number of subintervals within the piecewise linear function $\tilde{\bar{r}}\left(\tilde{\bar{P}};\lambda\right)$ maintains a constant slope. Since any conditional augmenting directed path $\tilde{\bar{P}}$ is an elementary path, results that $K\left(\tilde{\bar{P}}\right)\leq n-2$.

Theorem 2. (Conditional augmenting path theorem [9]) *A flow $\tilde{\bar{f}}$ is a parametric maximum flow if and only if the parametric residual network $\tilde{\bar{G}}\left(\tilde{\bar{f}}\right)$ contains no conditional augmenting directed path.*

4. Partitioning Algorithm for the Parametric Maximum Flow Problem

The partitioning algorithm (PA) for the parametric maximum flow problem presented in this paper determines in each of its iterations an improvement of the flow over a subinterval of the parameter values generated by the partition induced by the first (in increasing order of their λ values) of the breakpoints of the piecewise linear parametric residual capacity of the conditional augmenting directed paths $\tilde{\bar{P}}$ in the parametric residual network. Since the parametric residual capacities for all the arcs in $\tilde{\bar{G}}\left(\bar{f}\right)$ are linear functions of λ, the parametric residual capacity $\tilde{\bar{r}}\left(\tilde{\bar{P}};\lambda\right)$ of any conditional augmenting directed path $\tilde{\bar{P}}$ in the parametric residual network is a piecewise linear function of λ with $K\left(\tilde{\bar{P}}\right)$ breakpoints.

In order to avoid working with piecewise linear functions, the algorithm works in several parametric residual networks defined for subintervals of the parameter values where the parametric residual capacities of all arcs remain linear functions. The parametric residual network $\tilde{\bar{G}}\left(\bar{f}\right)$ defined for the subinterval $J_k=\left[\lambda_k,\lambda_{k+1}\right]$ of the parameter values is denoted by $\tilde{\bar{G}}_k\left(\bar{f}\right)$. Besides working with linear instead piecewise linear functions, another main advantage of our approach is that every augmenting directed path $\tilde{P}$ in a parametric residual network $\tilde{\bar{G}}_k\left(\bar{f}\right)$ is also a conditional augmenting directed path $\tilde{\bar{P}}$ in $\tilde{\bar{G}}\left(\bar{f}\right)$ for the subinterval $\tilde{I}\left(\tilde{\bar{P}}\right)=J_k$ for which the residual network $\tilde{\bar{G}}_k\left(\bar{f}\right)$ is defined.

The first phase of finding a parametric maximum flow consists in establishing a feasible flow, if one exists, in a non-parametric network $G^*=\left(N,A,\ell,u^*,s,t\right)$ obtained from the initial parametric network by only replacing the parametric upper bound functions with the non-parametric upper bounds: $u^*(i,j)=u_0(i,j)$ for $U(i,j)\geq 0$ and $u^*(i,j)=u_0(i,j)+\Lambda\cdot U(i,j)$ for $U(i,j)<0$. After a feasible flow f_0 is established, the next step is to compute the parametric residual network $\tilde{\bar{G}}(f_0)$ for this feasible flow. For the non-parametric flow f_0, the parametric residual capacities for every arc (i,j) in $\tilde{\bar{G}}(f_0)$ can be written as $\tilde{\bar{r}}(i,j;\lambda)=\tilde{\alpha}(i,j)+\lambda\cdot\tilde{\beta}(i,j)$, where $\tilde{\beta}(i,j)=U(i,j)$ represents the slope of the parametric residual capacity function and $\tilde{\alpha}(i,j)=u_0(i,j)-f_0(i,j)+f_0(j,i)-\ell(j,i)$ is the value of the parametric residual capacity function computed for $\lambda_0=0$, *i.e.* $\tilde{\bar{r}}(i,j;0)=\tilde{\alpha}(i,j)$.

The second phase of the algorithm starts with the parametric residual network $\tilde{\bar{G}}(f_0)$, defined for the non-parametric feasible flow f_0, which is also $\tilde{\bar{G}}_0(f_0)$, *i.e.* $\lambda_0=0$ and $J_0=[0,\Lambda]$, since the residual capacities of all arcs are linear functions. The algorithm repeatedly finds shortest augmenting directed paths from the source node to the sink node in the parametric residual network and increases the flow in the original parametric network $\bar{G}$ only in the subinterval $J_0=[0,\lambda_1]$ which reflects in updating the parametric residual network $\tilde{\bar{G}}_0(f)$. The parameter value λ_1 is updated on each flow augmentation step so that the parametric residual capacities of all arcs not to have breakpoints within the interval $J_0=[0,\lambda_1]$. During its successive iterations, the algorithm maintains an ordered list $B=\{\lambda_0=0,\lambda_1,\cdots,\lambda_k\}$ of parameter values for which the parametric network is partitioned. This list is initialised as $B:=\{0\}$ and is updated, each time the parametric residual network $\tilde{\bar{G}}_k\left(\bar{f}\right)$ contains no conditional augmenting directed path, with a new λ_{k+1} value, representing the new lower limit of the subinterval of the parameter values for which a new parametric residual network $\tilde{\bar{G}}_{k+1}\left(\bar{f}\right)$ is defined. At this point, the parametric maximum flow $\tilde{\bar{f}}_k$ is computed for the subinterval $J_k=\left[\lambda_k,\lambda_{k+1}\right]$ and the algorithm goes on iterating within the next subinterval $\left[\lambda_{k+1},\Lambda\right]$ until the value $\lambda_{k+1}=\Lambda$ is reached.

PARTITIONING ALGORITHM (PA);
1. BEGIN
2. compute a feasible flow f_0 in network G^*;
3. compute the parametric residual network $\tilde{\bar{G}}(f_0)$;
4. $B:=\{0\}$; $k:=0$; $\lambda_k:=0$;
5. REPEAT
6. SSADP(k,λ_k,B);
7. $k:=k+1$;

8. UNTIL ($\lambda_k = \Lambda$);
9. END.

In the *k*-th step of the partitioning algorithm (PA), the Successive Shortest Augmenting Directed Paths (SSADP) procedure computes the parametric residual network $\tilde{\tilde{G}}_k(f_0)$ for the subinterval $J_k = [\lambda_k, \Lambda]$, where the parametric residual capacities of all arcs can be written as $\tilde{\tilde{r}}_k(i,j;\lambda) = \alpha_k(i,j) + (\lambda - \lambda_k)\cdot\beta_k(i,j)$, with $\beta_k(i,j) = \tilde{\beta}(i,j)$ and $\alpha_k(i,j) = \tilde{\alpha}(i,j) + \lambda_k \cdot \tilde{\beta}(i,j)$. As can be easily seen, the restriction $\tilde{\tilde{r}}_k(i,j;\lambda) > 0$, $\forall\lambda \in (\lambda_k, \Lambda)$ is equivalent with $\alpha_k(i,j) + |\beta_k(i,j)| > 0$. The SSADP procedure maintains a partly conditional augmenting directed path $\tilde{\tilde{P}}(i)$ which is memorised in the predecessor vector π and executes ADVANCE and RETREAT operations from the current node i until the sink node t is reached, *i.e.* the partly conditional augmenting directed path is transformed in a conditional augmenting directed path $\tilde{\tilde{P}}$.

ADVANCE(i,j);
1. BEGIN
2. $\pi(j) := i$;
3. $i := j$;
4. END;

RETREAT(i);
1. BEGIN
2. $\tilde{d}(i) := \min\{\tilde{d}(j) + 1 | (i,j) \in \tilde{A}_k(\bar{f})\}$;
3. IF $i \neq s$ THEN $i := \pi(i)$;
4. END

From a current node i, an ADVANCE operation will add the admissible arc (i,j) to the partly conditional augmenting directed path while a RETREAT operation will eliminate the arc $(\pi(i), i)$ from it.

PROCEDURE SSADP(k, λ_k, B);
1. BEGIN
2. compute the parametrico residual network $\tilde{\tilde{G}}_k(f_0)$;
3. compute the exact distance labels $\tilde{d}(\cdot)$ in $\tilde{\tilde{G}}_k(f_0)$;
4. $\pi := \{n, n, \cdots, n\}$; $\alpha_k(\tilde{\tilde{P}}) := 0$; $\beta_k(\tilde{\tilde{P}}) := 0$; $i := s$; $\lambda_{k+1} := \Lambda$;
5. WHILE $\tilde{d}(s) < n$ DO
6. IF(exists an admissible arc (i,j)) THEN;
7. BEGIN
8. ADVANCE(i,j);
9. IF $(i = t)$ THEN
10. BEGIN
11. RC$(\pi, \lambda_{k+1}, B, \alpha_k(\tilde{\tilde{P}}), \beta_k(\tilde{\tilde{P}}))$;
12. $i := s$;
13. END;
14. END;
15. ELSE RETREAT(i);
16. compute the parametric maximum flow $\tilde{\tilde{f}}_k$;
17. add λ_{k+1} to the list B;
18. END;

A call to Residual Capacity (RC) procedure will compute the parametric residual capacity $\tilde{\tilde{r}}_k(\tilde{\tilde{P}};\lambda)$ of the conditional augmenting directed path and will update the values $\alpha_k(i,j)$ and $\beta_k(i,j)$ according to the augmentation of the flow. After initializing $\alpha_k(\tilde{\tilde{P}}) := \min\{\alpha_k(i,j) | (i,j) \in \tilde{\tilde{P}}\}$ and $\beta_k(\tilde{\tilde{P}}) := \min\{\beta_k(i,j) | (i,j) \in \tilde{\tilde{P}} \text{ and } \alpha_k(i,j) = \alpha_k(\tilde{\tilde{P}})\}$, in order to assure that the parametric residual capacity $\tilde{\tilde{r}}_k(\tilde{\tilde{P}};\lambda) = \alpha_k(\tilde{\tilde{P}}) + (\lambda - \lambda_k)\cdot\beta_k(\tilde{\tilde{P}})$ remains a linear function without breakpoints in the subinterval $(\lambda_k, \lambda_{k+1})$, the slope $\beta_k(\tilde{\tilde{P}})$ of the parametric residual capacity $\tilde{\tilde{r}}_k(\tilde{\tilde{P}};\lambda)$ is compared with the slopes $\beta_k(i,j)$ of the parametric residual capacities of all the arcs $(i,j) \in \tilde{\tilde{P}}$.

If the condition $\beta_k(i,j) < \beta_k(\tilde{\tilde{P}})$ holds for an arc $(i,j) \in \tilde{\tilde{P}}$, it means that the linear functions $\tilde{\tilde{r}}_k(\tilde{\tilde{P}};\lambda)$ and $\tilde{\tilde{r}}_k(i,j;\lambda)$ have a crossing point for a parameter value $\lambda^* > \lambda_k$ and, consequently, the parametric residual capacity $\tilde{\tilde{r}}_k(\tilde{\tilde{P}};\lambda)$ would have a breakpoint for

$$\lambda^* = \lambda_k + \left(\alpha_k(\pi(i),i) - \alpha_k(\tilde{\tilde{P}})\right) \Big/ \left(\beta_k(\tilde{\tilde{P}}) - \beta_k(\pi(i),i)\right).$$

If $\lambda^* < \lambda_{k+1}$, *i.e.* the breakpoint is placed within the subinterval $(\lambda_k, \lambda_{k+1})$, the upper limit λ_{k+1} will be replaced with the new parameter value λ^*. Then, the parametric residual capacity $\tilde{\tilde{r}}_k(\tilde{\tilde{P}};\lambda)$ will be subtracted from the parametric residual capacities of all arcs $(i,j) \in \tilde{\tilde{P}}$ and added to those of the arcs (j,i), for all the parameter values in the new subinterval $[\lambda_k, \lambda^*]$. As soon as $\tilde{\tilde{G}}_k(\bar{f})$ contains no conditional augmenting directed paths, the parametric maximum flow $\tilde{\tilde{f}}_k$ is computed for the subinterval $[\lambda_k, \lambda_{k+1}]$ and the value λ_{k+1} is added to the list B. Then the current value k of the counter is incremented and, if the condition $\lambda_k = \Lambda$ is not reached yet, the algorithm reiterates for the next subinterval $J_k = [\lambda_k, \lambda_{k+1}]$. Otherwise, if λ_k equals Λ, the whole interval of the parameter has been completed and the algorithm stops. For each of the subintervals $J_k = [\lambda_k, \lambda_{k+1}]$, $k := 0, 1, \cdots, K$ the parametric maximum flow is computed as

$$\tilde{\tilde{f}}_k(i,j;\lambda) := \ell(i,j) + \max\left\{\bar{u}(i,j;\lambda) - \tilde{\tilde{r}}_k(i,j;\lambda) - \ell(i,j), 0\right\}.$$

PROCEDURE RC$\left(\pi, \lambda_{k+1}, B, \alpha_k\left(\tilde{\tilde{P}}\right), \beta_k\left(\tilde{\tilde{P}}\right)\right)$;

1. BEGIN
2. compute $\left(\tilde{\tilde{P}}\right)$ based on predecessor vector π;
3. $\alpha_k\left(\tilde{\tilde{P}}\right) := \min\left\{\alpha_k(i,j) \middle| (i,j) \in \tilde{\tilde{P}}\right\}$;
4. $\beta_k\left(\tilde{\tilde{P}}\right) := \min\left\{\beta_k(i,j) \middle| (i,j) \in \tilde{\tilde{P}} \text{ and } \alpha_k(i,j) = \alpha_k\left(\tilde{\tilde{P}}\right)\right\}$;
5. $i := t$;
6. WHILE $i \neq s$ DO
7. BEGIN
8. IF $\beta_k(\pi(i), i) < \beta_k\left(\tilde{\tilde{P}}\right)$ THEN
9. BEGIN
10. $\lambda^* = \lambda_k + \left(\alpha_k(\pi(i), i) - \alpha_k\left(\tilde{\tilde{P}}\right)\right) \Big/ \left(\beta_k\left(\tilde{\tilde{P}}\right) - \beta_k(\pi(i), i)\right)$;
11. IF $\left(\lambda^* < \lambda_{k+1}\right)$ THEN $\lambda_{k+1} := \lambda^*$;
12. END;
13. $\alpha_k(\pi(i), i) := \alpha_k(\pi(i), i) - \alpha_k\left(\tilde{\tilde{P}}\right)$;
$\beta_k(\pi(i), i) := \beta_k(\pi(i), i) - \beta_k\left(\tilde{\tilde{P}}\right)$;
14. $\alpha_k(i, \pi(i)) := \alpha_k(i, \pi(i)) + \alpha_k\left(\tilde{\tilde{P}}\right)$;
$\beta_k(i, \pi(i)) := \beta_k(i, \pi(i)) + \beta_k\left(\tilde{\tilde{P}}\right)$;
15. $i := \pi(i)$;
16. END;
17. END;

Theorem 3. *The Successive Shortest Augmenting Directed Paths* (*SSADP*) *procedure correctly computes a parametric maximum flow in the parametric network* $\bar{G} = (N, A, \ell, \bar{u}, s, t)$ *for the parameter* λ *values in the subinterval* $J_k = [\lambda_k, \lambda_{k+1}]$.

Proof. Since procedure SSADP works in the parametric residual network $\tilde{\tilde{G}}_k\left(\bar{f}\right)$ for which the parametric residual capacities $\tilde{\tilde{r}}_k(i,j;\lambda)$ of all arcs $(i,j) \in \tilde{A}_k\left(\bar{f}\right)$ and the parametric residual capacity $\tilde{\tilde{r}}_k\left(\tilde{\tilde{P}};\lambda\right)$ of any of the augmenting directed paths $\tilde{\tilde{P}}$, are linear functions without crossing points within the subinterval $J_k = [\lambda_k, \lambda_{k+1}]$, the correctness of the procedure results from the correctness of the shortest augmenting directed path algorithm for the non-parametric case.

Theorem 4. *The Residual Capacity* (*RC*) *procedure correctly computes the parametric residual capacity* $\tilde{\tilde{r}}_k\left(\tilde{\tilde{P}};\lambda\right)$ *of a conditional augmenting directed path* $\tilde{\tilde{P}}$ *in the parametric residual network* $\tilde{\tilde{G}}_k\left(\bar{f}\right)$ *for the parameter* λ *values in the subinterval* $J_k = [\lambda_k, \lambda_{k+1}]$.

Proof. As the parametric residual capacity $\tilde{\tilde{r}}_k\left(\tilde{\tilde{P}};\lambda\right)$ is the inner envelope of the parametric residual capacity functions $\tilde{\tilde{r}}_k(i,j;\lambda)$ of all arcs composing the conditional augmenting directed path and since these parametric residual capacities are linear functions for the entire subinterval $J_k = [\lambda_k, \lambda_{k+1}]$, the proof results from choosing the minimum possible values (lines 3 and 4 of procedure RC) for $\alpha_k\left(\tilde{\tilde{P}}\right)$ and for the corresponding $\beta_k\left(\tilde{\tilde{P}}\right)$, as well as from continuously updating (line 11 of procedure RC) the upper limit λ_{k+1} of the subinterval for which the parametric residual network $\tilde{\tilde{G}}_k\left(\bar{f}\right)$ is defined.

Theorem 5. (Theorem of correctness) *If there is a feasible flow in the parametric network* $\bar{G} = (N, A, \ell, \bar{u}, s, t)$, *then the partitioning algorithm* (*PA*) *correctly computes a parametric maximum flow for* $\lambda \in [0, \Lambda]$.

Proof. The partitioning algorithm iterates on successive subintervals $J_k = [\lambda_k, \lambda_{k+1}]$, starting with $\lambda_0 = 0$ and ending with $\lambda_{k+1} = \Lambda$ and consequently, the correctness of the algorithm obviously follows from Theorem 3. Actually, the algorithm ends with a parametric maximum flow and with the partitioning of the interval of the parameter values: $J_k = [\lambda_k, \lambda_{k+1}]$, $k := 0, 1, \cdots, K$.

Theorem 6. (Theorem of complexity) *The partitioning algorithm* (*PA*) *for the parametric maximum flow problem runs in* $O(Kn^2m)$ *time, where* $K+1$ *is the number of* λ *values in the set* B *at the end of the algorithm.*

Proof. For each of the K subintervals $J_k = [\lambda_k, \lambda_{k+1}]$, $k := 0, 1, \cdots, K$ in which is partitioned the interval $[0, \Lambda]$ of the parameter values, the algorithm makes a call to procedure SSADP. Since the complexity of the procedure SSADP equals the complexity of the non-parametric successive shortest augmenting directed paths algorithm, being $O(n^2m)$, the total complexity of the partitioning algorithm is $O(Kn^2m)$.

5. Example

The algorithm is illustrated on the parametric network presented in **Figure 1** where the source node is $s = 0$, the sink node $t = 3$, and for the parameter λ taking values in the interval $[0,1]$, *i.e.* $\Lambda = 1$.

The feasible flow f_0, computed in the non-parametric network $G^* = (N, A, \ell, u^*, s, t)$, is presented in **Figure 2**,

the parametric residual network $\tilde{\tilde{G}}(f_0)$ is presented in **Figure 3** and the list B is initialised as $B := \{0\}$.

In the first iteration, for $k = 0$ and $\lambda_0 = 0$, the algorithm makes the first call to procedure SSADP which computes the parametric residual network $\tilde{\tilde{G}}_0(f_0)$. The values computed for $\alpha_0(i,j)$ and $\beta_0(i,j)$, as well as the exact distance labels $\tilde{d}(\cdot)$ in $\tilde{\tilde{G}}_0(f_0)$ are indicated in **Figure 4(a)**. The predecessor vector is initialized as $\pi := (4,4,4,4)$, $\alpha_0\left(\tilde{\tilde{P}}\right)$ and $\beta_0\left(\tilde{\tilde{P}}\right)$ are set to 0 and λ_1 is set to $\Lambda = 1$. The algorithm performs two consecutive ADVANCE steps over the admissible arcs (0,1) and respectively (1,3) and, since the sink node is reached, procedure RC is called which builds the conditional augmenting directed path $\tilde{\tilde{P}} = (0,1,3)$, based on the

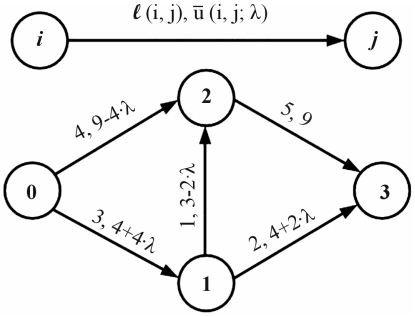

Figure 1. The parametric network $\bar{G} = (N, A, \ell, \bar{u}, s, t)$ with linear capacity functions and constant lower bounds.

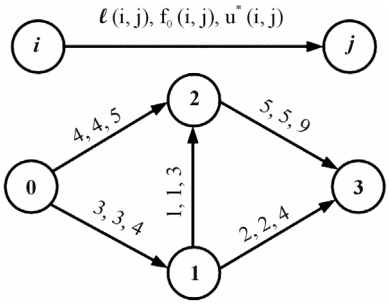

Figure 2. The feasible flow $f_0(i,j)$ in the non-parametric network $G^* = (N, A, \ell, u^*, s, t)$.

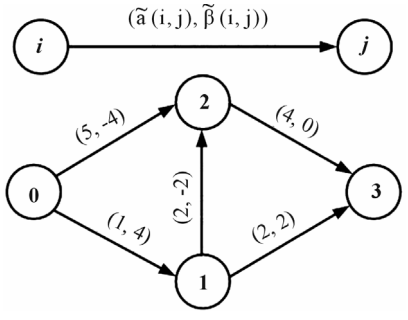

Figure 3. The parametric residual network $\tilde{\tilde{G}}(f_0)$.

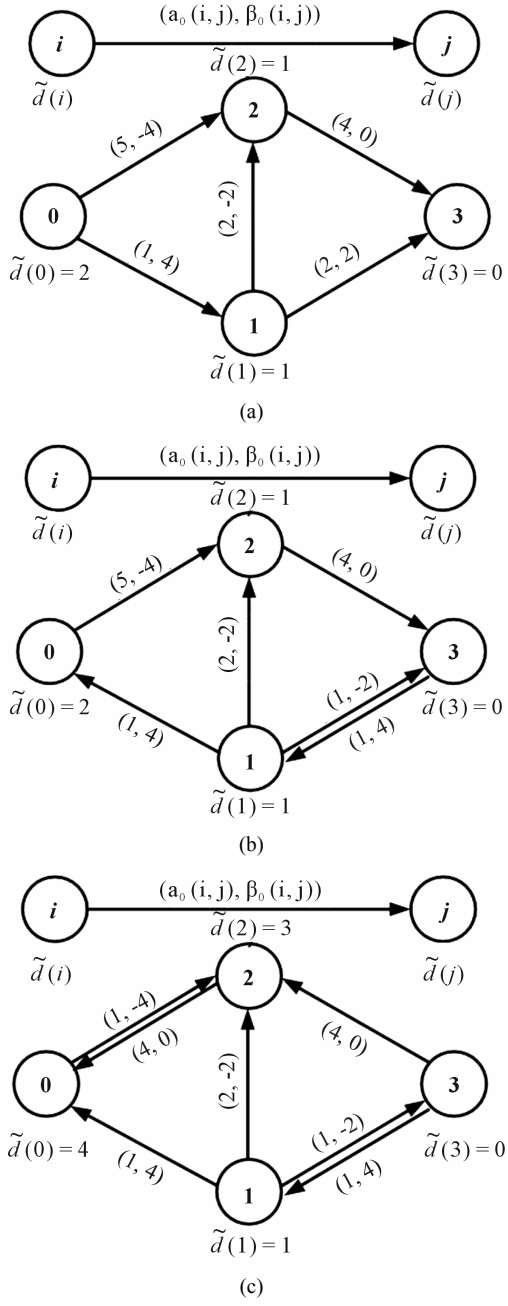

Figure 4. Exemplifying the first iteration performed by the partitioning algorithm (PA) for the parametric network $\bar{G}$ presented in Figure 1(a).

predecessor vector $\pi := (4,0,4,1)$, and computes the values $\alpha_0\left(\tilde{\tilde{P}}\right) = 1$ and $\beta_0\left(\tilde{\tilde{P}}\right) = 4$, *i.e.* the parametric residual capacity $\tilde{\tilde{r}}_0\left(\tilde{\tilde{P}}; \lambda\right) = 1 + 4\lambda$. The slope of the parametric residual capacity is compared with the slopes $\beta_0(i,j)$ of the parametric residual capacities of the arcs (1,3) and (0,1). Since the condition $\beta_0(1,3) = 2 < 4$ holds for the arc (1,3), the value $\lambda^* = 1/2$ is computed

and because $\lambda^* < \Lambda$, the upper limit of the subinterval of the parameter values is updated to $\lambda_1 := 1/2$. Then the values $\alpha_0(i,j)$ and $\beta_0(i,j)$ are updated for both arcs (1,3) and (0,1) and procedure RC ends with the parametric residual network $\tilde{\tilde{G}}_0(\tilde{\tilde{f}})$ presented in **Figure 4(b)**. Then, procedure SSADP makes two ADVANCE steps over the arcs (0,2) and (2,3) reaching again the sink node and procedure RC builds the new conditional augmenting directed path $\tilde{\tilde{P}} = (0,2,3)$ with the parametric residual capacity $\tilde{\tilde{r}}_0(\tilde{\tilde{P}};\lambda) = 4$, *i.e.* $\alpha_0(\tilde{\tilde{P}}) = 4$ and $\beta_0(\tilde{\tilde{P}}) = 0$. For the arc (0,2), the value $\lambda^* = 1/4$ is computed and since $\lambda^* = 1/4 < 1/2 = \lambda_1$ the upper limit of the subinterval J_0 is updated to $\lambda_1 := 1/4$.

Procedure SSADP selects again the admissible arc (0,2) and, since from node 2 there is no admissible arc, it is relabelled as $\tilde{d}(2) := \tilde{d}(0) + 1 = 3$ and a RETREAT step is performed to $i := \pi(2) = 0$. At this stage, there is no admissible arc in $\tilde{\tilde{G}}_0(\tilde{\tilde{f}})$ from the current node $i = 0$ and therefore, after relabeling node 0 as $\tilde{d}(0) := \tilde{d}(2) + 1 = 4$, this label does not meet the restriction $\tilde{d}(s) < n$. Based on the residual capacities presented in **Figure 4(c)**, the parametric flow $\tilde{\tilde{f}}_0$ (**Figure 5(a)**) is computed for the parameter values in the subinterval $J_0 = [\lambda_0, \lambda_1] = [0, 1/4]$ and the value $\lambda_1 = 1/4$ is added to the list B which becomes $B := \{0, 1/4\}$. After the procedure SSADP ends, the current value of the counter is incremented to $k := 1$ and, since $\lambda_1 \neq \Lambda$, a new iteration will be performed.

After performing three more iterations, the value $\lambda_4 = 1$ is added to the list B which becomes $B := \{0, 1/4, 1/2, 3/4, 1\}$ and the current value of the counter is incremented to $k := 4$. Since $\lambda_4 = \Lambda$, the partitioning algorithm ends. The parametric maximum flows computed by the algorithm are presented in **Figure 5**.

As can be noticed in **Figure 5**, the parametric maximum flow value function $\tilde{\tilde{v}}$ equals the capacity function $\tilde{\tilde{c}}[\tilde{S}_k; J_k] = \bar{u}(\tilde{S}_k, \tilde{T}_k) - \ell(\tilde{T}_k, \tilde{S}_k)$, with $J_k = [\lambda_k, \lambda_{k+1}]$, $k := 0,1,\cdots,K$, of the parametric minimum cut in the parametric network.

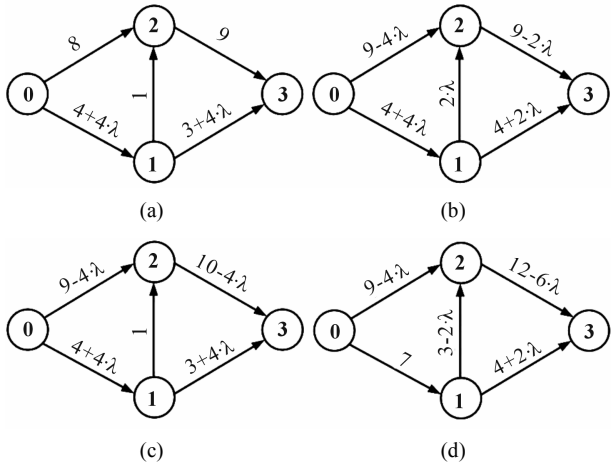

Figure 5. The parametric maximum flow for each of the subintervals J_k, k = 0, 1, 2, 3 of the parameter values: (a) J_0 = [0, 1/4]; (b) J_1 = [1/4, 1/2]; (c) J_2 = [1/2, 3/4]; (d) J_3 = [3/4, 1].

REFERENCES

[1] R. Ahuja, T. Magnanti and J. Orlin, "Network Flows. Theory, Algorithms and Applications," Prentice Hall Inc., Englewood Cliffs, 1993.

[2] H. W. Hamacher and L. R. Foulds, "Algorithms for Flows with Parametric Capacities," *ZOR—Methods and Models of Operations Research*, Vol. 33, No. 1, 1989, pp. 21-37.

[3] G. Ruhe, "Complexity Results for Multicriterial and Parametric Network Flows Using a Pathological Graph of Zadeh," *Zeitschrift für Operations Research*, Vol. 32, No. 1, 1988, pp. 9-27.

[4] G. Ruhe, "Characterization of All Optimal Solutions and Parametric Maximal Flows in Networks," *Optimization*, Vol. 16, No. 1, 1985, pp. 51-61. http://dx.doi.org/10.1080/02331938508842988

[5] C.-E. Bichot and P. Siarry, "Graph Partitioning: Optimisation and Applications," ISTE—Wiley, 2011.

[6] E. Ciurea and L. Ciupală, "About Preflow Algorithms for the Minimum Flow Problem," *WSEAS Transactions on Computer Research*, Vol. 1, No. 3, 2008, pp. 35-42.

[7] E. Ciurea and L. Ciupală, "Sequential and Parallel Algorithms for Minimum Flows," *Journal of Applied Mathematics and Computing*, Vol. 15, No. 1-2, 2004, pp. 53.E-75.E.

[8] G. Gallo, M. D. Grigoriadis and R. E. Tarjan, "A Fast Parametric Maximum Flow Algorithm and Applications," *SIAM Journal on Computing*, Vol. 18, No. 1, 1989, pp. 30-55. http://dx.doi.org/10.1137/0218003

[9] M. Parpalea, "Min-Max Algorithm for the Parametric Flow Problem," *Bulletin of the Transilvania University of Brasov*, *Series III*: *Mathematics*, *Informatics*, *Physics*, Vol. 3, No. 52, 2010, pp. 191-198.

