Minimum Flow in Monotone Parametric Bipartite Networks

Eleonor Ciurea, Mircea Parpalea

***Abstract*—** The algorithm presented in this paper solves the minimum flow problem for a special parametric bipartite network. The algorithm does not work directly in the original network but in the parametric residual network and finds a particular state of the residual network from which the minimum flow and the maximum cut for any of the parameter values are obtained. The approach implements a round-robin algorithm looping over a list of nodes until an entire pass ends without any change of the flow.

***Keywords*—** Balancing algorithm, Bipartite network, Minimum flow, Parametric flow.

I. Introduction

THE maximum flow problem is a fundamental problem in graph algorithms and optimization. Efficient algorithms for computing maximum flows are important not only because they are applied directly to the analysis of traffic or communication networks, but are often employed in the sub-problems of other network problems. Continuous improvements to algorithms have permanently been made by researchers for solving several classes of problems. Fundamental algorithms for network flow, including methods for maximum flow problem, were designed and efficient algorithms exist to solve different instances of this problem [1].

A natural generalization of the maximum flow problem is obtained by making the capacities of certain arcs functions of a single parameter. This problem is known as the parametric maximum flow problem. For the parametric maximum flow problem with linear capacity functions of a parameter λ Hamacher and Foulds [10] investigated an augmenting paths approach for determining in each iteration an improvement of the flow defined on the whole interval of the parameter. For the same problem, Ruhe [17], [18] proposed a "piece-by-piece" approach assuming that the maximum flow is known for a given value of the parameter and computing the maximum flow to be added to the current flow in order to preserve the maximally of the flow for greater parameter values and the maximum value of the parameter for which the computed flow is maximal. Gallo, Grigoriadis, and Tarjan [9] considered a special case of the parametric maximum flow problem in which the capacities of the arcs leaving the source are non-decreasing and the arcs entering the sink are non-increasing functions of the parameter while the capacities of all other arcs are constant. Although this type of parameterization appears to be quite specialized, Gallo et. al. [9] have pointed out that this parametric problem has many applications, in multiprocessor scheduling with release times and deadlines, integer programming problems, computing subgraph density and network vulnerability and partitioning a data base between fast and slow memory. For the same problem Ahuja et. al. [2], [3] proposed a parametric bipartite algorithm which finds all maximum flows for a given number of increasing values of the parameter. Recently Zhang, Tarjan et al. [20], [21], [22] proposed a balancing approach for the parametric maximum flow problem on bipartite networks. They refer to two algorithms, one of which uses two-arc augmenting paths and the other of which balances a star at each step, where a star is the subgraph induced by the set of arcs incident on a single vertex. The claimed main advantage of these algorithms is to be very simple as compared to other parametric maximum-flow algorithms.

Although it has its own applications, the minimum flow problem was not dealt so often as the maximum flow problem. Ciurea et.al. [4]-[8] investigated the non-parametric minimum flow problem. The parametric minimum flow problem has not been investigated so far by other authors. The parametric minimum flow problem is extension of the classical minimum flow problem in which the lower bounds of certain arcs are functions of a parameter λ. The parametric minimum flow problem is to compute all minimum flows for every possible value of the parameter. The minimum flow value function in a parametric network is a continuous piecewise linear function of the parameter. Each linear segment of the minimum flow value function between two breakpoints, λ_k and λ_{k+1}, corresponds to a cut that remains a maximum cut for any $\lambda_k < \lambda \leq \lambda_{k+1}$. The approach of the parametric minimum flow problem presented in this article refers to the minimum flow problem in a network with linear lower bound functions of a single parameter λ.

Further on, this paper is organized as follows: Section II contains the basic network flow terminology and some results used in the rest of the paper. More specialized terminology is developed in later sections. Section III deals with the parametric minimum flow problem, i.e. the minimum flow

Manuscript received July 8, 2010.

E. Ciurea, Department of Computer Science, *Transilvania* University of Brasov, 25, Eroilor, blvd, 500030, Brasov (e-mail: e.ciurea@ unitbv.ro).

M. Parpalea, National College *Andrei Saguna* 1, Saguna street, 500123, Brasov (e-mail: parpalea@gmail.com).

problem in a parametric network with linear lower bound functions. Section IV presents a balancing algorithm for the minimum flow problem in a special parametric bipartite network. Section V describes the steps performed by the former algorithm in a parametric bipartite network with lower bounds of the arcs leaving the source being non-increasing and the arcs entering the sink non-decreasing linear functions of a parameter while the capacities of all other arcs being constant. In the presentation to follow, some familiarity with flow algorithms is assumed and many details are omitted, since they are straightforward modifications of known results. The notions and results presented in Section II and Section III are taken from [1], [2], [4]-[8].

II. Terminology and Preliminaries

A. Definitions and Notation

Given a capacitated network $G=(N,A,\ell,u,s,t)$, let $n=|N|$ and $m=|A|$. The *upper bound* function and the *lower bound* function are two nonnegative functions, $u(i,j)$ and $\ell(i,j)$ associated with each arc $(j,j)\in A$. The network has two special nodes: a source node s and a sink node t. A *flow* is a function $f:A\rightarrow\Re^{+}$ satisfying the next conditions:

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

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:

$$\ell(i,j)\leq f(i,j)\leq u(i,j) \quad \forall(i,j)\in A \qquad (1.b)$$

for every arc $(i,j)\in A$ is referred to as a *feasible flow*. A *cut* is a partition of the node set N into two subsets S and $T=N-S$, represented using the notation $[S,T]$. Alternatively, a cut can be defined as the set of arcs whose endpoints belong to different subsets S and T. A cut is nontrivial if both S and T are nonempty. An arc (i,j) with $i\in S$ and $j\in T$ is referred to as a *forward arc* of the cut while an arc (i,j) 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 let (T,S) denote the set of backward arcs. A cut $[S,T]$ is a *s-t cut* if $s\in S$ and $t\in T$.

A network $G=(N,A,\ell,u,s,t)$ is called *bipartite* if its node set N can be partitioned into two subsets N_1 and N_2 such that all arcs have one endpoint in N_1 and the other in N_2. Let $n_1=|N_1|$ and $n_2=|N_2|$. A bipartite network is often represented using the notation $G=(N_1\bigcup N_2,A,\ell,u,s,t)$. Let source s and sink t be two distinguished nodes in the bipartite network, considering that $s\in N_2$ and $t\in N_1$.

B. The Minimum Flow Problem

The minimum flow problem is to determine a flow $\hat{f}$ for which v is minimized. By convention, if the arc $(i,j)\in A$ and the arc $(j,i)\notin A$, the arc (j,i) is added to the set of arcs A by setting $\ell(j,i)=0$ and $u(j,i)=0$. Let f be a feasible solution for the minimum flow problem. Supposing that an arc $(i,j)\in A$ carries $f(i,j)$ units of flow, the existing flow can be decreased either by *pulling* $f(i,j)-\ell(i,j)$ units of flow from j to i over the arc (i,j) or by *pushing* $u(j,i)-f(j,i)$ units of flow from j to i along the arc (j,i).

The *residual capacity* $\hat{r}(i,j)$ of any arc $(i,j)\in A$, with respect to a given flow f, is given by:

$$\hat{r}(i,j)=u(j,i)-f(j,i)+f(i,j)-\ell(i,j). \qquad (2)$$

Based on these ideas, for a network $G=(N,A,\ell,u,s,t)$ and a feasible solution f, the network denoted by $\hat{G}(f)=(N,\hat{A})$, where $\hat{A}$ is the set of *residual arcs* corresponding to the feasible solution f, consisting only of arcs (i,j) with $\hat{r}(i,j)>0$ is referred to as the *residual network* with respect to the given flow f for the minimum flow problem. From residual capacities, a flow can be determined using the following expression:

$$f(i,j)=\ell(i,j)+\max\{\hat{r}(i,j)-u(j,i)+\ell(j,i),\ 0\}. \qquad (3)$$

The *capacity* of a $s-t$ cut $\hat{c}[S,T]$ is defined, for the minimum flow problem, as the sum of the lower bounds of the forward arcs minus the sum of the upper bounds of the backward arcs:

$$\hat{c}[S,T]=\ell(S,T)-u(T,S). \qquad (4)$$

The *s-t* cut with the greatest capacity value among all *s-t* cuts is referred to as a *maximum cut,* $[\hat{S},\hat{T}]$.

Theorem 1 (*Min-Flow Max-Cut Theorem*): *If there is a feasible flow in the network, the value of the minimum flow from a source s to a sink t in a capacitated network with nonnegative lower bounds equals the capacity of the maximum $s-t$ cut.*

In the residual network $\hat{G}(f)=(N,\hat{A})$ the *distance function,* $\hat{d}:N\to\aleph$ is a function from the set of nodes to the nonnegative integers. A distance function is said to be *valid* if it satisfies the following conditions:

$$\hat{d}(s)=0 \text{ and } \hat{d}(j)\le\hat{d}(i)+1,\ \forall\ (i,j)\in\hat{A}. \tag{5}$$

The value $\hat{d}(i)$ is referred to as the *distance label* of node i. The distance labels are said to be *exact* if for each node i, $\hat{d}(i)$ equals the length of the shortest path from node s to node i in the residual network $\hat{G}(f)$.

Definition 1: An arc (i,j) in the residual network $\hat{G}(f)$ is referred to as *admissible* if it satisfies the condition: $\hat{d}(j)=\hat{d}(i)+1$; otherwise it is *inadmissible*. A directed path from the source node s to the sink node t in the residual network $\hat{G}(f)$ is called *admissible directed path* if it only consists of admissible arcs; otherwise it is *inadmissible*.

Definition 2: A directed path $\hat{P}(i)$, from the source node s to a node i ($i\neq t$) in the residual network $\hat{G}(f)$, consisting only of admissible arcs is called *partly admissible directed path*.

Statement 1: *An admissible directed path from the source node s to the sink node t in the residual network $\hat{G}(f)$ is a shortest decreasing directed path.*

Statement 2:

(*a*) *If the distance labels are valid, the distance label $\hat{d}(i)$ is a lower bound on the length of the shortest directed path from node s to node i in the residual network.*

(*b*) *If $\hat{d}(t)\ge n$, the residual network contains no directed path from the source node s to the sink node t.*

Theorem 2 (*Decreasing Path Theorem*): *A flow f is a minimum flow if and only if the residual network $\hat{G}(f)$ contains no directed path from the source node to the sink node.*

Definition 3: For the minimum flow problem, a *preflow* is a function $f:A\to\Re^{+}$ satisfying the next conditions:

$$\sum_{j|(i,j)\in A}f(i,j)-\sum_{j|(j,i)\in A}f(j,i)\le 0,\ \textit{for all } i\in N-\{s,t\} \tag{6.a}$$

$$\ell(i,j)\le f(i,j)\le u(i,j)\quad\forall(i,j)\in A \tag{6.b}$$

Definition 4: The *deficit* of each node $i\in N$ is defined as:

$$\hat{e}(i)=\sum_{j|(i,j)\in A}f(i,j)-\sum_{j|(j,i)\in A}f(j,i), \tag{7}$$

thus, for the minimum flow problem, for any preflow f, $\hat{e}(i)\le 0$, $i\in N-\{s,t\}$.

A node $i\in N\setminus\{s,t\}$ is said to be *active* if $\hat{e}(i)<0$ and *balanced* if $\hat{e}(i)=0$. A preflow f for which $\hat{e}(i)=0$, $i\in N-\{s,t\}$ is a flow. Consequently, a flow is a particular case of preflow.

There are three approaches for solving minimum flow problem:

- using decreasing directed path algorithms from source node s to sink node t in residual network $\hat{G}(f)$;
- using preflow-pull algorithms starting from sink node t in residual network $\hat{G}(f)$;
- using augmenting directed path algorithms from sink node t to source node s or using preflow-push algorithms starting from sink node t in residual network $\tilde{G}(f)$, where $\tilde{G}(f)$ is the residual network defined for the maximum flow.

III. Flows in Parametric Networks

For the parametric flow problem, the lower bound and upper bound functions as well as the flow functions on any of the arcs (i,j) are piecewise linear functions defined on an interval $[0,\Lambda]$. On the set of all piecewise linear functions $f(\lambda)$ an ordering cannot be defined for the entire interval $[0,\Lambda]$ since two piecewise linear functions are not necessarily comparable. Therefore a partitioning J_k of the interval of the parameter $[0,\Lambda]$ into disjoints subintervals $J_1\cup\ldots\cup J_K=[0,\Lambda]$ with $J_p\cap J_q=\phi$, $\forall\ p\neq q$ must be defined so that on each of the subintervals J_k an ordering to be defined as:

$$f_1(\lambda)\le f_2(\lambda)\ \text{for}\ J_k\ \Leftrightarrow\ f_1(\lambda)\le f_2(\lambda),\ \forall\,\lambda\in J_k. \tag{8}$$

For the parametric minimum flow problem a network $G(\lambda)=(N,A,\ell(\lambda),u,s,t)$ is considered, where the lower bound function $\ell(i,j;\lambda)$ of each arc $(i,j)\in A$ is a linear function of a single, non-negative, real parameter, λ:

$$\ell(i,j;\lambda)=\ell_0(i,j)-\lambda\cdot £(i,j). \tag{9}$$

The real valued function, $£(i,j)$ associated with each arc, $(i,j)\in A$ is referred to as the *parametric part of the lower bound* of the arc (i,j). The non-negative value $\ell_0(i,j)$ is the lower bound of the arc (i,j) for $\lambda=0$: $\ell(i,j;0)=\ell_0(i,j)$ and $0\le\ell_0(i,j)\le u(i,j)$. The parameter λ takes values in

the interval $[0,\Lambda]$ where Λ is chosen so that: $0 \le \ell(i,j;\lambda) \le u(i,j)$, $\forall (i,j) \in A$; $\forall \lambda \in [0,\Lambda]$. Therefore the parametric part of the lower bounds, $£(i,j)$ satisfy the constraints: $(\ell_0(i,j) - u(i,j))/\Lambda \le £(i,j) \le \ell_0(i,j)/\Lambda$, $\forall (i,j) \in A$..

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

$$\textit{minimize} \quad v(\lambda) \quad \textit{for all} \quad \lambda \in [0,\Lambda] \tag{10}$$

with

$$\sum_{j|(i,j)\in A} f(i,j;\lambda) - \sum_{j|(j,i)\in A} f(j,i;\lambda) = \begin{cases} v(\lambda), & i = s \\ 0, & i \ne s,t \\ -v(\lambda), & i = t \end{cases} \tag{11.a}$$

$$\ell(i,j;\lambda) \le f(i,j;\lambda) \le u(i,j) \quad \forall (i,j) \in A. \tag{11.b}$$

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

Let $f(\lambda) = (\ldots f(i,j;\lambda), \ldots)_{(i,j)\in A}$ be a vector of flow functions defined on the interval $[0,\Lambda]$. Supposing that an arc $(i,j) \in A$ carries a flow $f(i,j;\lambda)$, the existing flow can be reduced either by *pulling* the flow $f(i,j;\lambda) - \ell(i,j;\lambda)$ from node j to node i over the arc (i,j) or by *pushing* the flow $u(j,i) - f(j,i;\lambda)$ from j to i along the arc (j,i). These flows are computed as differences between piecewise linear functions of λ. The *parametric residual capacity* $\hat{r}(i,j;\lambda)$ of any arc $(i,j) \in A$, with respect to a given flow $f(\lambda)$, is given by:

$$\hat{r}(i,j;\lambda) = u(j,i) - f(j,i;\lambda) + f(i,j;\lambda) - \ell(i,j;\lambda). \tag{12}$$

For a network $G(\lambda) = (N, A, \ell(\lambda), u, s, t)$ and a feasible solution $f(\lambda)$, the network denoted by $\hat{G}(\lambda, f) = (\hat{N}, \hat{A})$, with $\hat{N} = N$ and $\hat{A}$ being the set of arcs consisting only of arcs with $\hat{r}(i,j;\lambda) > 0$ for at least a subinterval of $[0,\Lambda]$, is referred to as the *parametric residual network* with respect to the given flow $f(\lambda)$ for the parametric minimum flow problem. From the parametric residual capacities $\hat{r}(i,j;\lambda)$, the flow can be determined using the following expression:

$$f(i,j;\lambda) = \ell(i,j;\lambda) + \max\{\hat{r}(i,j;\lambda) - u(j,i) + \ell(j,i;\lambda), 0\}. \tag{13}$$

Definition 5: *A parametric cut partitioning* $[S_k; J_k]$ is a finite set of cuts $[S_k, T_k]$, $k = 1,\ldots,K$ together with a partitioning J_k of the interval of the parameter $[0,\Lambda]$ into disjoints subintervals so that $J_1 \cup \ldots \cup J_K = [0,\Lambda]$ and $J_p \cap J_q = \phi$, $\forall p \ne q$. The capacity of a parametric $s-t$ cut partitioning for the minimum flow problem is a piecewise linear function $\hat{c}[S_k; J_k]$ defined for all λ of every subinterval $\lambda \in J_k$, $k = 1,\ldots,K$:

$$\hat{c}[S_k; J_k] = \sum_{(i,j)\in(S_k,T_k)} \ell(i,j;\lambda) - \sum_{(i,j)\in(T_k,S_k)} u(i,j). \tag{14}$$

A parametric $s-t$ cut for which the subintervals of the parameter values $\hat{J}_k$ assure that every $s-t$ cut is a maximum cut $[\hat{S}_k; \hat{T}_k]$ for all $\lambda \in \hat{J}_k$ is referred to as a *parametric maximum* $s-t$ *cut,* $[\hat{S}_k; \hat{J}_k]$ for the whole interval of the parameter values, $[0,\Lambda]$. Thus a parametric maximum cut $[\hat{S}_k; \hat{J}_k]$ is a set of maximum cuts $[\hat{S}_k; \hat{T}_k]$ and $\hat{c}[\hat{S}_k; \hat{J}_k] = \hat{c}[\hat{S}_k; \hat{T}_k]$ for all λ of every subinterval $\hat{J}_k$, $k = 1,\ldots,K$.

Theorem 3 (*Parametric Min-Flow Max-Cut Theorem*): *If there is a feasible flow in the parametric network, the value function of the parametric minimum flow from a source* s *to a sink* t *in a capacitated network with parametric lower bounds equals the capacity of the parametric maximum* $s-t$ *cut.*

Proof: From the non-parametric Min-Flow Max-Cut Theorem (*Theorem 1*) results that for any of the parameter values λ^*, the value of the non-parametric minimum flow with respect to fixed lower bounds $\ell(i,j;\lambda^*)$, $f(\lambda^*) = (\ldots f(i,j;\lambda^*), \ldots)_{(i,j)\in A}$ equals the capacity of the maximum $s-t$ cut: $\hat{v}(\lambda^*) = \hat{c}[\hat{S}, \hat{T}]^*$. From the definition of the subintervals of the parameter values $\hat{J}_k$ for the parametric maximum $s-t$ cut results that if $\lambda^* \in \hat{J}_k$ then the maximum $s-t$ cut $[\hat{S}, \hat{T}]^*$ remains a maximum cut $[\hat{S}_k, \hat{T}_k]$ for the entire subinterval $\hat{J}_k$. Hence, for every subinterval $\hat{J}_k$ holds that $\hat{v}(\lambda) = \hat{c}[\hat{S}_k, \hat{T}_k]$, for all $\lambda \in J_k$ and since $\hat{c}[\hat{S}_k; \hat{J}_k] = \hat{c}[\hat{S}_k; \hat{T}_k]$ for all λ of every subinterval $\hat{J}_k$, $k = 1,\ldots,K$ results that the value function of the parametric minimum flow from the source node s to the sink node t equals the capacity of the parametric maximum $s-t$ cut: $\hat{v}(\lambda) = \hat{c}[\hat{S}_k; \hat{J}_k]$, $k = 1,\ldots,K$ for all $\lambda \in [0,\Lambda]$. ■

The subintervals:

$$\hat{I}(i,j) = \{\lambda \mid \hat{r}(i,j;\lambda) > 0\} \quad for\ (i,j) \in \hat{A} \tag{15}$$

describe subintervals of $[0,\Lambda]$, $\hat{I}(i,j) \subseteq [0,\Lambda]$ where a decreasing of flow along an arc (i,j) in $\hat{G}(\lambda,f)$ is possible, based on $f(\lambda)$. If an arc (i,j) doesn't belong to $\hat{G}(\lambda,f)$ then $\hat{I}(i,j) := \phi$ is set.

Definition 6: A *conditional decreasing directed path* $\hat{P}(\lambda)$ in $\hat{G}(\lambda,f)$ is a directed path $\hat{P}$ from the source node s to the sink node t such that: $\hat{I}(\hat{P}) = \bigcap_{(i,j)\in\hat{P}} \hat{I}(i,j) \neq \phi$.

Definition 7: The parametric residual capacity of a conditional decreasing directed path $\hat{P}(\lambda)$ in $\hat{G}(\lambda,f)$ is the inner envelope of the parametric residual capacities, $\hat{r}(i,j;\lambda)$ of all arcs composing the conditional decreasing directed path for all $\lambda \in \hat{I}(\hat{P})$:

$$\hat{r}(\hat{P},\lambda) = \min_{\lambda\in\hat{I}(\hat{P})} \{\hat{r}(i,j;\lambda) \mid (i,j) \in \hat{P}(\lambda)\}. \tag{16}$$

Theorem 4 (*Conditional Decreasing Path Theorem*): *A flow* $f(\lambda)$ *is a parametric minimum flow if and only if the parametric residual network* $\hat{G}(\lambda,f)$ *contains no conditional decreasing directed path from the source node to the sink node.*

Proof: If $f(\lambda)$ is a parametric minimum flow then there does not exist any $\lambda^* \in [0,\Lambda]$ so that a decreasing directed path $\hat{P}(\lambda^*)$ from the source node to the sink node in the non-parametric residual network $\hat{G}(\lambda^*,f)$ can be found, otherwise $f(\lambda^*)$ would not be a minimum flow. Thus, if $f(\lambda)$ is a parametric minimum flow then the parametric residual network $\hat{G}(\lambda,f)$ contains no conditional decreasing directed path from the source node to the sink node.

Mutually, suppose that no conditional decreasing directed path exists in $\hat{G}(\lambda,f)$ and yet $f(\lambda)$ is not a parametric minimum flow, i.e. there exists $\lambda^* \in [0,\Lambda]$ such that $v(\lambda^*) > \hat{v}(\lambda^*)$ where $v(\lambda^*)$ is the value of the flow $f(\lambda^*)$. Consequently, $f(\lambda^*) = (\ldots f(i,j;\lambda^*)\ldots)_{(i,j)\in A}$ is therefore a feasible but not minimum flow with respect to the fixed lower bounds $\ell(i,j;\lambda^*)$. Hence, by the *decreasing path theorem* for non-parametric flows, there exists a directed path $\hat{P}^*$ in $\hat{G}(\lambda^*,f)$ with respect to $f(\lambda^*)$. The definition of $\hat{G}(\lambda,f)$ yields that $\hat{P}^*$ is also a conditional decreasing directed path $\hat{P}(\lambda)$ in $\hat{G}(\lambda,f)$ with $\hat{I}(\hat{P}) = \{\lambda^*\}$ which contradicts the non-existence of a conditional decreasing directed path. Thus, if no conditional decreasing directed path from the source node to the sink node exists in the parametric residual network $\hat{G}(\lambda,f)$ then $f(\lambda)$ is a parametric minimum flow. ■

IV. Minimum Flow Balancing Algorithm for a Monotone Parametric Bipartite Network

For the parametric maximum flow problem, Zhang, Tarjan et al. [20], [21], [22] proposed a balancing approach for the parametric maximum flow problem on bipartite networks. They refer to two algorithms, one of which uses arc balancing and the other of which balances a star at each step, where a star is the subgraph induced by the set of arcs incident on a single vertex. The claimed main advantage of these algorithms is to be very simple as compared to other parametric maximum-flow algorithms. Gallo et. al. [9] showed that if the parameterized bounds are linear functions of λ, the maximum flow value function of λ is a piecewise linear function with no more than $n-2$ breakpoints.

The *balancing algorithm for the minimum flow in a parametric bipartite network* decreases the flow over simple decreasing directed path in a special parametric bipartite network with linear lower bound functions of the parameter for all the arcs leaving the source or entering the sink while all other arcs having constant lower bounds. For the parametric minimum flow problem a flow is referred to as λ-*balanced* if there is no simple decreasing directed path from the source node to the sink node in the parametric residual network.

A. Monotone Parametric Bipartite Network

The *monotone parametric bipartite network* for the minimum flow problem is a special kind of parametric bipartite network with the source connected to all nodes $i \in N_1$ with $\{(i,s)\} = \phi$ and all nodes $j \in N_2$ connected to the sink with $\{(t,j)\} = \phi$. The lower bounds of the arcs out of the source are non-increasing functions of a parameter λ and the lower bounds of the arcs into the sink are non-decreasing functions of λ, the lower bounds of the remaining arcs being constant. In the following, the parametric lower bounds are linear functions of a parameter λ taking values in the interval $[0,\Lambda]$, that is:

$$\ell(s,i;\lambda) = \ell_0(s,i) - \lambda \cdot £(s,i) \quad with$$
$$0 \le £(s,i) \le \frac{1}{\Lambda}\ell_0(s,i), \quad \forall\,(s,i) \in A, \tag{17}$$

$$\ell(j,t;\lambda) = \ell_0(j,t) - \lambda \cdot £(j,t) \quad with$$

$$\frac{1}{\Lambda}(\ell_0(j,t)-u(j,t))\le £(j,t)\le 0,\ \ \forall (j,t)\in A. \quad (18)$$

B. Balancing Algorithm

For a feasible flow $f(\lambda)$, if for an arc out of the source, (s,i) holds that $f(s,i;\lambda)\ge \ell(s,i;\lambda)\ \ \forall \lambda\in[0,\Lambda]$ then the flow can be decreased along the arc (s,i) for the entire interval of the parameter. Let a step k of flow decreasing be considered starting with a feasible flow $f_k(\lambda)$. For every arc (s,i) the value λ_i^k is defined to be the maximum value of λ for which $f_k(s,i;\lambda)=\ell(s,i;\lambda)$, i.e. $\lambda_i^k := max\{\lambda \mid \hat{r}(s,i;\lambda)=0\}$. Equivalently, for every arc (j,t), the value λ_j^k is defined to be the minimum value of λ such that $f_k(j,t;\lambda)=\ell(j,t;\lambda)$, i.e. $\lambda_j^k := min\{\lambda \mid \hat{r}(j,t;\lambda)=0\}$. Actually for a certain value λ_i^k the flow can be decreased along the arc (s,i) only for the parameter values $\lambda\in(\lambda_i,\Lambda]$ since the lower bounds of the arcs out of the source are non-increasing functions of λ. Similarly, for a value λ_j^k defined for an arc entering the sink the flow can be decreased along the arc (j,t) only for the parameter values $\lambda\in[0,\lambda_j)$ since the lower bounds of the arcs into the sink are non-decreasing functions of λ. Consequently the flow can be decreased along a conditional decreasing directed path including the arcs (s,i) and (j,t) as long as $[0,\lambda_j^k)\bigcap(\lambda_i^k,\Lambda]\ne\phi$, i.e. the flow is always pulled along a directed path containing arcs (i,j) out of a node i with a smaller lambda value and ending in a node j with a greater lambda value, $\lambda_i^k<\lambda_j^k$. This situation is illustrated in Fig. 1. According to the general definition a flow $f(\lambda)$ is said to be λ-*balanced* if there is no decreasing path avoiding the source s from a node i to a node j with $\lambda_i<\lambda_j$.

The idea is that a λ-balanced flow is a minimum flow which, to the extent possible, equalizes the flows on the arcs out of the source s, where equalization is with respect to λ-values. For the monotone parametric bipartite network in discussion, a flow is referred to as λ-*balanced* if there is no decreasing directed path, or even an arc with positive residual capacity, from a node i with lower lambda value to a node j with higher lambda value. After a step of decreasing of flow along a directed path containing an arc (i,j), the new subinterval available for flow decreasing reduces to ϕ if not constrained by the residual capacity of the arc (i,j).

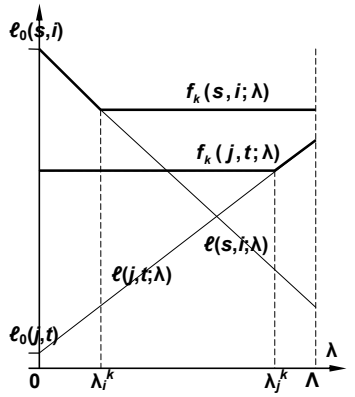

Fig. 1 Lower bound functions for arcs (s,i) and (j,t) in G(λ)

C. Decreasing the Flow along Simple Directed Paths in Residual Network

The algorithm works in the parametric residual network, $\hat{G}(\lambda,f)=(N,\hat{A})$.

Definition 8: A directed path $\hat{P}=(s,i,j,t)$ with $i\in N_1-\{t\}$ and $j\in N_2-\{s\}$ in $\hat{G}(\lambda,f)=(N,\hat{A})$ is referred to as a *simple residual path* if one of the following two cases holds:

a) $\lambda_i<\lambda_j$ *and* $\hat{r}(i,j)>0$ *for* $\lambda\in(\lambda_i,\lambda_j)$

b) $\lambda_i>\lambda_j$ *and* $\hat{r}(j,i)>0$ *for* $\lambda\in(\lambda_j,\lambda_i)$.

Let λ_i^k and λ_j^k denote the lambda values at the beginning of a k step of decreasing, respectively λ_i^{k+1}, λ_j^{k+1} the corresponding values at the end of the decreasing step over a simple residual path, $\hat{P}=(s,i,j,t)$ in the parametric residual network $\hat{G}(\lambda,f_k)$.

For the case when $\lambda_i^k<\lambda_j^k$, assuming not being constrained by the residual capacity of the arc (i,j), a pull of a flow from t to s will reduce the subinterval available for decreasing to ϕ, i.e. $\lambda_i^{k+1}=\lambda_j^{k+1}=\lambda^*$, which is illustrated in Fig 2.

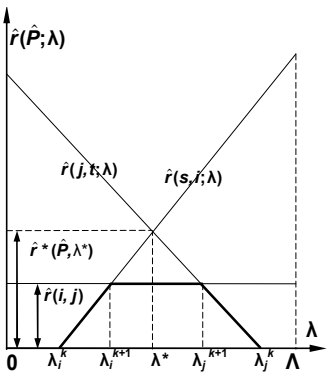

Fig. 2 Decreasing the flow over a Simple Residual Path with $\lambda_i^k<\lambda_j^k$

The maximum value $\hat{r}^*(\hat{P},\lambda^*)$ of the piecewise linear residual capacity function for which the two new lambda values become equal after the pull is computed as: $\hat{r}^*(\hat{P},\lambda^*)=\hat{r}(s,i;\lambda^*)=\hat{r}(j,t;\lambda^*)$. Otherwise, the decreasing of flow will make the two new lambda values $\lambda_i^{k+1},\lambda_j^{k+1}$ come closer to each other. These new values are computed as solutions of the following equations: $\hat{r}(s,i;\lambda_i^{k+1})=\hat{r}(i,j)$ and $\hat{r}(j,t;\lambda_j^{k+1})=\hat{r}(i,j)$. For this case, after the pull of flow, the residual capacity of the arc (i,j) will become zero for all the vales of the parameter: $\lambda\in(\lambda_i^{k+1},\lambda_j^{k+1})$. Hence, a pull of flow equal to $\hat{r}(\hat{P},\lambda)$ is performed from the sink node t to the source node s through the directed path $\hat{P}(\lambda)$ in $\hat{G}(\lambda,f_k)$.

For the second case, when $\lambda_i^k>\lambda_j^k$, a pull of $\hat{r}^*(\hat{P}_r,\lambda)$ flow is performed from the source node s to the sink node t through the reversed directed path $\hat{P}_r(\lambda)$ in $\hat{G}(\lambda,f_k)$ for the two lambda values to become equal. If the residual capacity of the arc (j,i) in $\hat{G}(\lambda,f_k)$ restricts the pull of the flow by being smaller then the maximum value of the piecewise linear residual capacity $\hat{r}^*(\hat{P}_r,\lambda^*)$, i.e. $\hat{r}(j,i;\lambda)<\hat{r}^*(\hat{P}_r,\lambda^*)$ for $\lambda\in(\lambda_j^k,\lambda_i^k)$, then the two new lambda values $\lambda_i^{k+1},\lambda_j^{k+1}$ come closer to each other being computed as solutions of the following equations: $\ell(s,i;\lambda_i^{k+1})=\ell(s,i;\lambda_i^k)+\hat{r}(j,i)$ and $\ell(j,t;\lambda_j^{k+1})=\ell(j,t;\lambda_j^k)+\hat{r}(j,i)$. Thus, for every arc $(i,j)\in\hat{A}$:

a) If $\lambda_i^k<\lambda_j^k$, a pull of $\hat{r}(\hat{P},\lambda)$ flow from the sink node t to the source node s through the directed path $\hat{P}(\lambda)$ in $\hat{G}(\lambda,f_k)$ is performed;

b) If $\lambda_i^k>\lambda_j^k$, a pull of $\hat{r}(\bar{P}_r,\lambda)$ flow from the source node s to the sink node t through the reversed directed path $\hat{P}_r(\lambda)$ in $\hat{G}(\lambda,f_k)$ is performed.

For the lambda values at the end of a decreasing step holds that:

a) if $\lambda_i^k<\lambda_j^k$, then $\lambda_i^k<\lambda_i^{k+1}\le\lambda_j^{k+1}<\lambda_j^k$ and

b) if $\lambda_i^k>\lambda_j^k$, then $\lambda_i^k>\lambda_i^{k+1}\ge\lambda_j^{k+1}>\lambda_j^k$.

After each step of decreasing the flow either the residual capacity of the arc (i,j) becomes zero or the two new lambda values become equals. A single operation will never reverse the order of the two lambda values even though the order could be reversed in the future by other operations. The flow to be pulled, $\hat{r}(\hat{P},\lambda)$ along the directed path $\hat{P}(\lambda)=(s,i,j,t)$ and $\hat{r}(\hat{P}_r,\lambda)$ along the reversed directed path $\hat{P}_r(\lambda)=(t,j,i,s)$ are computed as follows:

$$\hat{r}(\hat{P},\lambda):=\begin{cases}0 & for\ \lambda\in[0,\lambda_i^k]\\ \hat{r}(s,i;\lambda) & for\ \lambda\in(\lambda_i^k,\lambda_i^{k+1}]\\ \hat{r}(i,j;\lambda) & for\ \lambda\in(\lambda_i^{k+1},\lambda_j^{k+1})\\ \hat{r}(j,t;\lambda) & for\ \lambda\in[\lambda_j^{k+1},\lambda_j^k)\\ 0 & for\ \lambda\in[\lambda_j^k,\Lambda]\end{cases}, \tag{19}$$

$$\hat{r}(\hat{P}_r,\lambda):=\begin{cases}0 & for\ \lambda\in[0,\lambda_j^k]\\ \ell(j,t;\lambda)-\ell(j,t;\lambda_j^k) & for\ \lambda\in(\lambda_j^k,\lambda_j^{k+1}]\\ \hat{r}(j,i;\lambda) & for\ \lambda\in(\lambda_j^{k+1},\lambda_i^{k+1})\\ \ell(s,i;\lambda)-\ell(s,i;\lambda_i^k) & for\ \lambda\in[\lambda_i^{k+1},\lambda_i^k)\\ 0 & for\ \lambda\in[\lambda_i^k,\Lambda]\end{cases} \tag{20}$$

D. Balancing Algorithm for the Monotone Parametric Bipartite Network (BMPB)

The first phase of finding a parametric minimum flow consists in establishing a feasible flow in a non-parametric network $G'=(N,A,\ell',u,s,t)$ which is obtained from the initial network $G(\lambda)=(N,A,\ell(\lambda),u,s,t)$ by modifying the parametric lower bounds as follows:

$$\ell'(s,i)=max\{\ell(s,i;\lambda)\,|\,\lambda\in[0,\Lambda]\},$$
$$\text{i.e. } \ell'(s,i)=\ell_0(s,i) \tag{21}$$

and

$$\ell'(j,t)=max\{\ell(j,t;\lambda)\,|\,\lambda\in[0,\Lambda]\},$$
$$\text{i.e. } \ell'(j,t)=\ell_0(j,t)-\Lambda\cdot u(j,t). \tag{22}$$

The second phase consists in finding the parametric minimum flow and the complete set of maximum cuts for the entire interval of the parameter values, i.e. the parametric maximum cut. The algorithm maintains a set L of nodes $j\in N_2-\{s\}$ whose last examination resulted in a change in the flow. Initially $L:=N_2-\{s\}$ and during a pass over L if an examination of a node j results in sufficiently small flow change or no flow change, j is dropped from L. When L becomes empty it is reset to $L:=N_2-\{s\}$. If an entire pass over $L:=N_2-\{s\}$ results in no flow change the computation

is complete and a parametric minimum flow $\hat{f}(\lambda)$ is obtained. From the lambda values, λ_j the maximum cut of the original network $G(\lambda)=(N,A,\ell(\lambda),u,s,t)$ under any λ value is derived in a single linear scan of the nodes.

```
(1)  Programme BMPB;
(2)  Begin
(3)    find an initial feasible flow f in G;
       compute the parametric residual
(4)    network Ĝ(λ,f);
(5)    for i:=1 to n1 do λi=0;
(6)    for j:=1 to n2 do λj=Λ;
(7)    L0:=Ø;
(8)    for j:=1 to n2 do add node j to L0;
(9)    L:=L0;
(10)    while L≠Ø do
(11)      begin
(12)        B:=0;
(13)        remove the first node j from L;
(14)        select the first arc (i,j)
            adjacent to node j with i≠t;
(15)        repeat
(16)          C:=0;
(17)          if ((r̂(i,j)>0)and(λi<λj)) then
(18)            begin
(19)              pull r̂(P̂,λ) over P̂(λ)=(s,i,j,t);
(20)              update λi and λj;
(21)              update Ĝ(λ,f);
(22)              B:=1;
(23)            end
(24)          else if((r̂(j,i)>0)and(λi>λj))then
(25)            begin
(26)              pull r̂(P̂r,λ) over P̂r(λ)=(t,j,i,s);
(27)              update λi and λj;
(28)              update Ĝ(λ,f);
(29)              B:=1;
(30)            end;
(31)          if (i,j) is not the last arc
              adjacent to j then
(32)            begin
(33)              select the next arc (i,j)
                  adjacent to node j with i≠t;
(34)              C:=1;
(35)            end;
(36)        until(C=0);
(37)        if (B=1) then add node j to L;
(38)        if ((L=Ø)and(B=1)) then L:=L0;
(39)      end;
(40) End.
```

Theorem 5: *If there is a feasible flow in the network* $G(\lambda)=(N,A,\ell(\lambda),u,s,t)$, *the* BMPB *algorithm computes correctly a parametric minimum flow.*

Proof: When the algorithm terminates, there is no arc with positive residual capacity in the parametric residual network, from any node i with lower lambda value to any node j with higher lambda value. For any of the parameter values $\lambda_k \in (0,\Lambda]$ the nodes in $N-\{s,t\}$ are partitioned according to their lambda values in two disjoints sets S_k and T_k. Let S_k be the set of nodes $i \in N-\{s,t\}$ with $\lambda_i < \lambda_k$, $S_k := \{i \mid \lambda_i < \lambda_k\}$ and $T_k := \{j \mid \lambda_j \geq \lambda_k\}$ which means that the two sets generates the $s-t$ cut $[S_k,T_k]$. For every arc $(i,j) \in (S_k,T_k)$ the residual capacity of the arc $\hat{r}(i,j;\lambda)=0$ since $\lambda_i < \lambda_j$, i.e. $f(i,j;\lambda)-\ell(i,j;\lambda)=0$. Equivalently for every arc $(j,i) \in (T_k,S_k)$ the residual capacity of the arc $\hat{r}(j,i;\lambda)=0$ since $\lambda_i < \lambda_j$, i.e. $u(j,i)-f(j,i;\lambda)=0$. Consequently the flow can not be decreased over the $s-t$ cut $[S_k,T_k]$ since the residual capacity of any of the simple directed paths from the source node to the sink node $\hat{r}(\hat{P},\lambda)=0$ and the residual capacity of any of the simple reversed directed paths from the sink node to the source node $\hat{r}(\hat{P}_r,\lambda)=0$. Hence for all lambda values the flow $f(\lambda)=(\ldots f(i,j;\lambda),\ldots)_{(i,j)\in A}$ is a parametric minimum flow. ■

E. Deriving the Parametric Maximum Cut

Using the residual network $\hat{G}(\lambda,\hat{f})$ with the parametric minimum flow given by the balancing algorithm, a maximum cut of the parametric network $G(\lambda)=(N,A,\ell(\lambda),u,s,t)$ under any λ value is derived in a single linear scan of the nodes. The set of nodes N is partitioned in two disjoint subsets $[S_k, T_k]$ as follows:

$$S_k=\{s\}\bigcup\{i \mid \lambda_i < \lambda_k\}\bigcup\{j \mid \lambda_j < \lambda_k\} \text{ and } T_k = N - S_k. \quad (23)$$

The complete procedure for getting the parametric maximum cut is summarized as follows:

```
(1)  procedure max-cut;
(2)  begin
(3)     N:=N-{s,t};
(4)     λ1:=min{λi | i∈N}
(5)     S1:={s};
(6)     J1:=[0,λ1];
(7)     N1:={i∈N | λi=λ1}
(8)     N:=N-N1;
(9)     k:=1;
(10)    repeat
(11)          λk:=min{λi | i∈N}
(12)          Sk:= Sk-1∪Nk-1;
(13)          Jk:=(λk-1,λk];
(14)          Nk:={i∈N | λi=λk}
(15)          N:=N-Nk;
(16)          k:=k+1;
(17)    until(N=Ø);
(18)    Sk:= Sk-1∪Nk-1;
(19)    Jk:=(λk-1,Λ];
(20) end;
```

F. Complexity Issues

The approach implements a round-robin algorithm which consists in looping over a fixed list of nodes and performing a decrease over each simple each simple residual path until an entire pass over the list results no decreasing. It is easy to construct examples on which this algorithm runs forever. Therefore, a suitable stopping condition can be introduced.

It suffices to stop the algorithm from iterating when nodes joined by a simple decreasing directed path have λ-values close enough that a simple post processing phase will complete the computation. For instance, nodes i and j joined by a residual decreasing directed path have λ-values close enough, $|\lambda_j - \lambda_i| \le \varepsilon$.

Theorem 6: *The ε-approximation of the BMPB algorithm has the complexity of* $O(n_1 \cdot n_2 \log(\Lambda/\varepsilon))$.

Proof: The flow is decreased over an arc (i,j) in $\hat{G}(\lambda, f)$ with positive residual capacity only if $\lambda_i + \varepsilon < \lambda_j$. Pulling flow over all arcs (i,j) entering in the same node j results in each of the iterations in λ_j-values dropping by a factor, given by the parametric part of the lower bounds of the parameterized arcs in the simple decreasing directed path containing node j. The number of iterations of the main loop is $O(\log(R))$, where R is the ratio between the initial difference between λ-values, Λ and the minimum possible difference between λ-values, e.g. $R := \Lambda/\varepsilon$. The algorithm stops when a pass over the arcs, taking $O(n_1 \cdot n_2)$ time, ends without any change of the flow. Thus, the algorithm stops after $O(\log(\Lambda/\varepsilon))$ passes over the arcs, taking $O(n_1 \cdot n_2)$ time per pass, for a total time of $O(n_1 \cdot n_2 \log(\Lambda/\varepsilon))$. ■

Although this algorithm is not competitive with other algorithms, it has the advantage of being extremely simple and intuitive.

For the purpose of finding the parametric maximum cut and the minimum flow value function a single scan of the nodes N in the increasing order of their associated λ-values gives all maximum cuts as monotone increasing step set-function S_k of λ and the λ_k-values, at which the sizes of the maximum cut partitions change, i.e. the partitioning J_k of the interval of the parameter, $[0,\Lambda]$. Sorting the nodes by their associated lambda value takes $O(n\log(n))$ time. All minimum flow value functions of the original network can be calculated in $O(m)$ time, where *m* is the number of arcs, if the arcs are also scanned along with the scanning of the nodes to get the capacities of the max-cuts. Since the maximum cuts are nested, each arc is scanned twice – once when one of its ends changes membership from T_k partition to S_k partition and the second time when the other end changes its membership. Let X_k denote the set of nodes which change membership from T_k to S_{k+1} while passing from $[S_k, J_k]$ partition to $[S_{k+1}, J_{k+1}]$ partition. Since $S_{k+1} = S_k \cup X_k$ and $T_{k+1} = T_k \setminus X_k$, follows that:

$$(S_{k+1}, T_{k+1}) = (S_k, T_{k+1}) \cup (X_k, T_{k+1}) = (S_k, T_k) \cup (X_k, T_{k+1}) \setminus (S_k, X_k)$$

and

$$(T_{k+1}, S_{k+1}) = (T_{k+1}, S_k) \cup (T_{k+1}, X_k) = (T_k, S_k) \cup (T_{k+1}, X_k) \setminus (X_k, S_k).$$

Thus, the recursive relation for computing the capacity of the parametric maximum cut is:

$$\hat{c}[\hat{S}_{k+1}; \hat{J}_{k+1}] = \sum_{(i,j)\in(\hat{S}_{k+1}, \hat{T}_{k+1})} \ell(i,j;\lambda) - \sum_{(i,j)\in(\hat{T}_{k+1}, \hat{S}_{k+1})} u(i,j) =$$

$$= \sum_{(i,j)\in(\hat{S}_k, \hat{T}_k)} \ell(i,j;\lambda) - \sum_{(i,j)\in(\hat{T}_k, \hat{S}_k)} u(i,j) + \sum_{(i,j)\in(X_k, \hat{T}_{k+1})} \ell(i,j;\lambda) -$$

$$- \sum_{(i,j)\in(\hat{T}_{k+1}, X_k)} u(i,j) - \sum_{(i,j)\in(\hat{S}_k, X_k)} \ell(i,j;\lambda) + \sum_{(i,j)\in(X_k, \hat{S}_k)} u(i,j) =$$

$$= \hat{c}[\hat{S}_k; \hat{J}_k] - \left(\sum_{(i,j)\in(\hat{S}_k, X_k)} \ell(i,j;\lambda) - \sum_{(i,j)\in(X_k, \hat{S}_k)} u(i,j) \right) +$$

$$+ \left(\sum_{(i,j)\in(X_k, \hat{T}_{k+1})} \ell(i,j;\lambda) - \sum_{(i,j)\in(\hat{T}_{k+1}, X_k)} u(i,j) \right).$$

V. EXAMPLE

The algorithm is illustrated on the network presented in Fig. 3. The feasible flow in the network is illustrated in Fig. 4. and the parametric residual network for the initial feasible flow f_0 with the parameter λ taking values in the interval [0,1] is presented in Fig. 5. In any of the parametric residual network representations, for every arc (i,j), the parametric residual capacity $\hat{r}(i,j;\lambda) > 0$ is indicated only for the values of the parameter $\lambda \in [\lambda_i, \lambda_j]$.

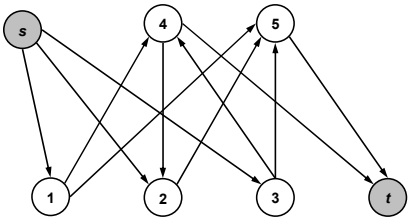

Fig. 3 The bipartite network $G(\lambda)$

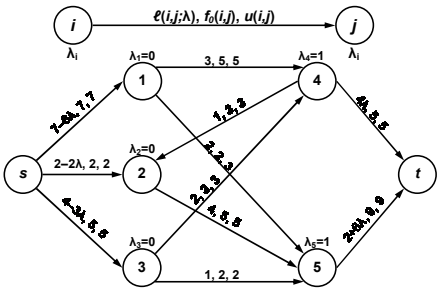

Fig. 4 The feasible flow f_0 in network $G(\lambda)$

***Iteration* 1**: From the list $L=\{4,5\}$, node 4 is removed and the first adjacent arc $(1,4)$ is selected. Since $\lambda_1<\lambda_4$ and $\hat{r}(1,4;\lambda)>0$, the simple residual path $(s,1,4,t)$ is found. Computing the value for $\hat{r}*(s,1,4,t;\lambda^*)$, $\hat{r}*(s,1,4,t;\lambda^*)=\hat{r}(s,1;\lambda^*)=\hat{r}(4,t;\lambda^*)$, results that $\lambda^*=1/2$ and $\hat{r}(s,1,4,t;\lambda^*)=3$.

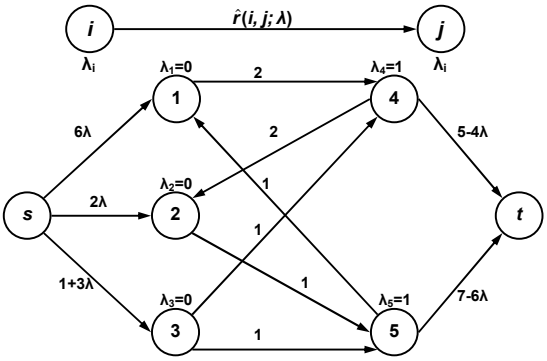

Fig. 5 The parametric residual network $\hat{G}(\lambda, f_0)$

Since $\hat{r}*(s,1,4,t;\lambda^*)>\hat{r}(1,4;\lambda)$ a pull of $\hat{r}(\hat{P},\lambda)$ flow, indicated in table 1, is performed along the residual directed path $\hat{P}=(s,1,4,t)$ from the sink node to the source node. The new lambda values, $\lambda_1=1/3$ and $\lambda_4=3/4$ are computed as $\hat{r}(s,1;\lambda_1)=\hat{r}(1,4;\lambda_1)$ and $\hat{r}(4,t;\lambda_4)=\hat{r}(1,4;\lambda_4)$ respectively. The updated residual network is indicated in Fig. 6.

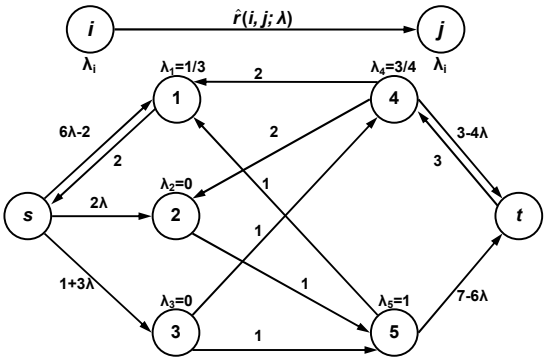

Fig. 6 The updated parametric residual network after the first iteration

Since the arc $(1,4)$ is not the last arc adjacent to node 4, the next arc is selected: arc $(2,4)$. Since $\hat{r}(2,4;\lambda)=0$ the algorithm does not iterates for this arc and the next arc is selected.

***Iteration* 2**: For the arc $(3,4)$, $\lambda_3<\lambda_4$ and $\hat{r}(3,4;\lambda)>0$, the simple residual path $(s,3,4,t)$ is found for which the values for $\hat{r}*(s,3,4,t;\lambda^*)=\hat{r}(s,3;\lambda^*)=\hat{r}(4,t;\lambda^*)$ are computed, resulting in $\lambda^*=2/7$ and $\hat{r}(s,3,4,t;\lambda^*)=13/7$.

The value for $\hat{r}*(s,3,4,t;\lambda^*)>\hat{r}(1,4;\lambda)$ and therefore a pull of $\hat{r}(\hat{P},\lambda)$ flow, indicated in table 1 is performed along the residual directed path $\hat{P}=(s,3,4,t)$ from the sink node to the source node.

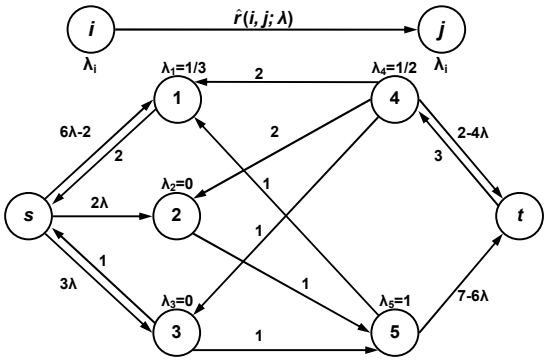

Fig. 7 The updated parametric residual network after the second iteration

The new lambda values, $\lambda_3=0$ and $\lambda_4=1/2$ are computed and the updated residual network is indicated in Fig. 7. At this stage of the algorithm there are no more arcs adjacent to node 4 and therefore node 4 is added to the list L.

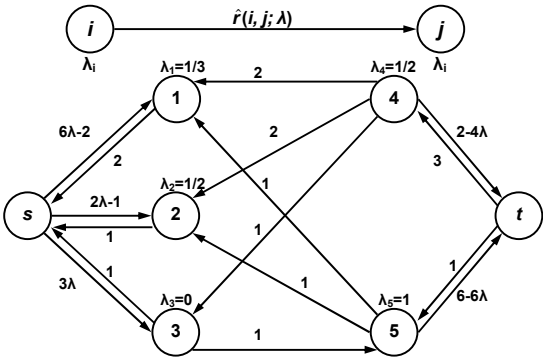

Fig. 8 The updated parametric residual network after the third iteration

***Iteration* 3**: Node 5 is removed from the list and the first adjacent arc $(1,5)$ is selected. Since $\lambda_1<\lambda_5$ but $\hat{r}(1,5;\lambda)=0$, the algorithm selects the next arc.

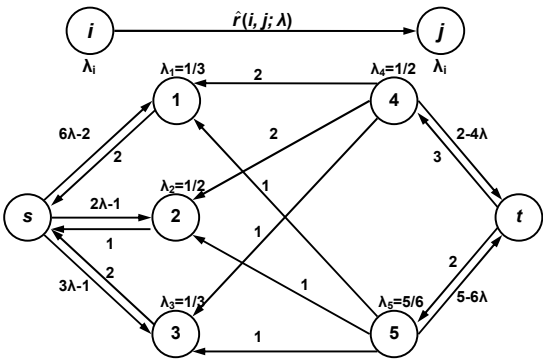

Fig. 9 The updated parametric residual network after the forth iteration

For the arc $(2,5)$, the simple residual path $(s,2,5,t)$ is found and a pull of $\hat{r}(\hat{P},\lambda)$ flow, with $\hat{r}(\hat{P},\lambda)$ indicated in Table 1, is performed from the sink node to the source node. The new lambda values are: $\lambda_2=1/2$ and $\lambda_5=1$. The updated residual network is presented in Fig. 8.

(i,j)	λ_i^k	λ_j^k	$\hat{r}(\hat{P},\lambda)$	λ_i^{k+1}	λ_j^{k+1}
(1,4)	0	1	$6\cdot\lambda \quad \lambda\in[0,1/3]$ $2 \quad \lambda\in(1/3,\ 3/4)$ $5-4\cdot\lambda \quad \lambda\in[3/4,1]$	1/3	3/4
(3,4)	0	3/4	$1 \quad \lambda\in[0,1/2]$ $3-4\cdot\lambda \quad \lambda\in(1/2,\ 3/4)$ $0 \quad \lambda\in[3/4,1]$	0	1/2
(2,5)	0	1	$2\cdot\lambda \quad \lambda\in[0,1/2]$ $1 \quad \lambda\in(1/2,1]$	1/2	1
(3,5)	0	1	$3\cdot\lambda \quad \lambda\in[0,1/3]$ $1 \quad \lambda\in(1/3,\ 5/6)$ $6-6\cdot\lambda \quad \lambda\in[5/6,1]$	1/3	5/6

Table I The four iterations of the BMPB algorithm

***Iteration* 4**: In the residual network in figure 4.b, the algorithm selects the last simple residual path $\hat{P}=(s,3,5,t)$ with the residual capacity indicated in table 1. Node 5 is added to the list L, then the two nodes are removed consecutively and all arcs are investigated without finding any simple residual paths. The updated residual network $\hat{G}(\lambda,f)$ after the last iteration in Fig. 9 has no simple residual paths and initial network with the parametric minimum flow is presented in Fig. 10.

In order of deriving maximum cuts and minimum flow value functions of the original parametric network, the nodes in $N-\{s,t\}$ are sorted by their associated lambda value in an increasing order as presented in Fig. 11.

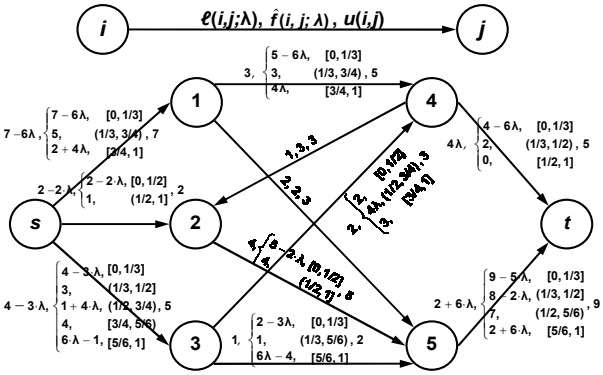

Fig. 10 The initial network with the parametric minimum flow

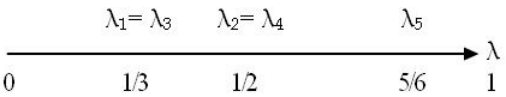

Fig. 11 Nodes in $N-\{s,t\}$ sorted by their associated lambda values in an increasing order

In Fig. 12, the maximum cuts for the original network are presented. For every subinterval J_k of $[0,\Lambda]$ resulting from the λ_k-values, at which the sizes of the maximum cut partitions change, the parametric minimum flow value function equals the capacity of the maximum cut, $\hat{v}(\lambda)=\hat{c}[\hat{S}_k;\hat{J}_k]$.

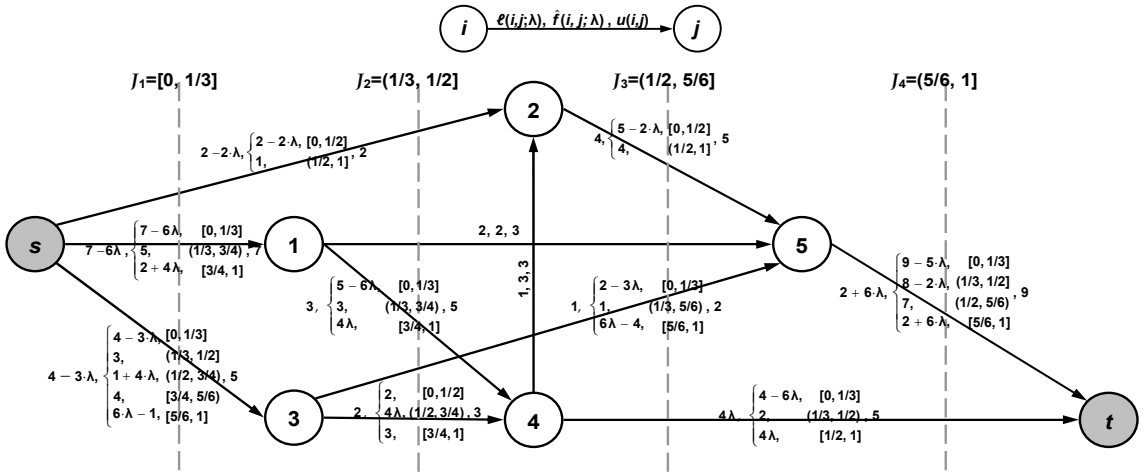

Fig. 12 The original parametric network with the nodes $N-\{s,t\}$ sorted in the increasing order of their associated λ values. (The maximum cuts and the flow values for every arc are indicated)

The piecewise linear minimum flow value function for the monotone parametric bipartite network with breakpoints for all λ_k values computed by the balancing algorithm is presented in Fig. 13

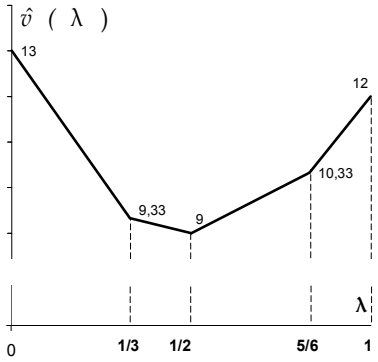

Fig. 13 The piecewise minimum flow value function for the parametric bipartite network in Fig. 3

REFERENCES

[1] Ahuja,R., Magnanti, T., Orlin, J. (1993): Network Flows. Theory, algorithms and applications. Prentice Hall, Inc., Englewood Cliffs, New Jersey.

[2] Ahuja,R., Stein,C., Tarjan,R.E., Orlin, J.(1994): Improved algorithms for bipartite network flow. SIAM *Journal on Computing*, 23 (5), pp.906-933.

[3] Ahuja,R., Orlin, J. (1990): Distance-Directed Augmenting Path Algorithms for Maximum Flow and Parametric Maximum Flow Problems. Naval Research Logistics, 38, pp.413-430.

[4] Ciurea, E., Ciupală, L. (2008): About preflow algorithms for the minimum flow problem. WSEAS *Transactions on Computer Research*, Issue 1, vol.3, pp.35-42

[5] Ciurea, E., Ciupală, L. (2008): A Parallel Algorithm for the minimum flow problem in bipartite networks. WSEAS *International Conference on Computers*, Heraklion, Grece, pp.203-207.

[6] Ciurea, E., Ciupală, L. (2008): Decreasing path algorithm for the minimum flows. Dynamic tree implementations. WSEAS *International Conference on Computers*, Heraklion, Grece, pp.235-240.

[7] Ciurea, E., Ciupală, L. (2008): Sequential and Parallel Deficit Scaling Algorithms for Minimum Flow in Bipartite Networks. WSEAS *Transactions on Computers*, Issue 10, vol.7, pp.1545-1554.

[8] Ciurea, E., Ciupală, L. (2007): A Highest-Label Preflow Algorithm for the Minimum Flow Problem. *Proceedings of the* 11th WSEAS *International Conference on Computers*, Agios Nikolaos, Crete Island, Greece, pp.168-171.

[9] Gallo, G., Grigoriadis, M. D. and Tarjan, R. E. (1989): A fast parametric maximum flow algorithm and applications. SIAM *Journal on Computing* 18, pp.30-55.

[10] Harnacher, H.W., L. R. Foulds, L.R.(1989): Algorithms for Flows with Parametric Capacities. ZOR - *Methods and Models of Operations Research,* Volume 33, pp.21-37

[11] Karp, R., Orlin, J. (1981): Parametric Shortest Path algoritthms with an Application to Cyclic Staffing. Discrete Applied Mathematics 3, pp.37-45. North-Holland Publishing Company

[12] Kolmogorov, V., Boykov, Y., Rother, C. (2007): Applications of parametric maxflow in computer vision. IEEE 11th *International Conference on In Computer Vision,* pp.1-8.

[13] Lu, Ch., Chen, Y. (2006): Using Multi-Thread Technology Realize Most Short-Path Parallel Algorithm. *World Academy of Science, Engineering and Technology* 15, pp.11-13

[14] Parbery, I. (1987): Parallel Complexity Theory. Pitman Publishing, London.

[15] Ruhe, G. (1991): Algorithmic Aspects of Flows in Networks. *Kluwer Academic Publishers*, Dordrecht.

[16] Ruhe, G. (1988): Parametric Maximal Flows in Generalized Networks– Complexity and Algorithms. Optimization, *Journal of Mathematical Programming and Operations Research*, vol.19, Issue 2, pp.235–251.

[17] Ruhe, G. (1988): Complexity Results for Multicriterial and Parametric Network Flows Using a Pathological Graph of Zadeh. *Zeitschrift für Operations Research*, Volume 32. pp.9- 27.

[18] Ruhe, G. (1985): Characterization of all optimal solutions and parametric maximal flows in networks. *Optimization*, vol.16, Issue 1, pp.51-61.

[19] Scutellà, M.G. (2007): A note on the parametric maximum flow problem and some related reoptimization issues. *Annals of Operations Research. Combinatorial Optimization and Application*, 150, pp.231–244.

[20] Tarjan, R., Ward, J., Zhang, B., Zhou, Y., Mao, J. (2006): Balancing Applied to Maximum Network Flow Problems. Algorithms – ESA, *Springer* 4168/2006, pp.612-623.

[21] Zang, B. (2007): A New Balancing Method for Solving Parametric Maximum Flow Problems. Stanford EE Computer Systems Colloquium.

[22] Zang, B., Ward, J., Feng, Q. (2005): A Simultaneous Parametric Maximum-Flow Algorithm with Vertex Balancing. *Technical Report* HPL-2005-121, HP Laboratories, Palo Alto.

[23] Zang, B., Ward, J., Feng, Q. (2004): A Simultaneous Parametric Maximum-Flow Algorithm for Finding the Complete Chain of Solutions. *Technical Report* HPL-2004-189, HP Laboratories, Palo Alto.