Balancing Algorithm for the Minimum Flow Problem in Parametric Bipartite Networks

ELEONOR CIUREA¹, MIRCEA PARPALEA²
¹Department of Computer Science
Transilvania University of Brasov
25, Eroilor, blvd, 500030, Brasov
²National College “Andrei Saguna”
1, Saguna, street, 500123, Brasov
ROMANIA

e.ciurea@unitbv.ro, parpalea@gmail.com, http://www.unitbv.ro

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 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.

Key-Words: Parametric minimum flow, bipartite network, balancing algorithm

1 Introduction

The parametric maximum flow problem with linear capacity functions of a parameter \( \lambda \) was investigated by Hamacher and Foulds [5] and Ruhe [6]. Gallo, Grigoriadis, and Tarjan [4] and Zhang, Tarjan et al. [7], [8] considered the special case of a monotone parametric maximum flow problem. The approach of the parametric minimum flow problem presented in this article refers to the minimum flow problem in a monotone parametric network with linear lower bound functions of a single parameter \( \lambda \). In the presentation to follow, some familiarity with flow algorithms is assumed and many details are omitted. The notions and results presented in Section 2 are taken from [1], [2] and [3].

2 Terminology and preliminaries

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 \( f:A \to \mathbb{R}^+ \) satisfying the next conditions:

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

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:

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

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 \( j \in S \) is referred to 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 \( G \) 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 denoted by the notation \( G=(N_1 \cup N_2,A,\ell,u,s,t) \).

Let source \( s \) and sink \( t \) be two distinguished nodes in the bipartite network, considering that \( s \in N_1 \) and \( t \in N_1 \). Also, \( (i,s) \) := \( \phi \) and \( (t,j) \) := \( \phi \) is considered.

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

\[
\hat{r}(i,j) = u(j,i) - f(j,i) + f(i,j) - \ell(i,j).
\]  

(3)

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:

\[
f(i,j) = \ell(i,j) + \max\{\hat{r}(i,j) - u(j,i) + \ell(i,j), 0\}. \tag{4}
\]

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:

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

(5)

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.

**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.

### 3 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 \( I_1 \) of the interval of the parameter \([0, \Lambda]\) into disjoints subintervals \( I_1 \bigcup \ldots \bigcup I_k = [0, \Lambda] \) with \( I_p \bigcap I_q = \phi \) \( \forall p \neq q \) must be defined so that on each of the subintervals \( I_k \) an ordering to be defined as:

\[
f_1(\Lambda) \leq f_2(\Lambda) \text{ for the subinterval } I_k \iff \quad f_1(\Lambda) \leq f_2(\Lambda), \quad \forall \Lambda \in I_k.
\]  

(6)

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, nonnegative, real parameter, \( \Lambda \):

\[
\ell(i,j;\Lambda) = \ell_0(i,j) - \Lambda \ell(i,j).
\]  

(7)

The real valued function, \( \ell(i,j) \) 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 \( \Lambda = 0 \):

\[
\ell(i,j;0) = \ell_0(i,j) \text{ and } 0 \leq \ell_0(i,j) \leq u(i,j).
\]

The parameter \( \Lambda \) takes values in the interval \([0, \Lambda]\) where \( \Lambda \) is chosen so that: \( 0 \leq \ell(i,j;\Lambda) \leq u(i,j) \), \( \forall (i,j) \in A \). Therefore the parametric part of the lower bounds, \( \ell(i,j) \) satisfy the constraints:

\[
\frac{1}{\Lambda} (\ell_0(i,j) - u(i,j)) \leq \ell(i,j) \leq \frac{1}{\Lambda} \ell_0(i,j), \quad \forall (i,j) \in A.
\]

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

\[
\text{minimize } \nu(\Lambda) \text{ for all } \Lambda \in [0, \Lambda]
\]  

(8)

with

\[
\sum_{(j,j) \in A} f(i,j;\Lambda) - \sum_{(j,j) \in A} f(j,i;\Lambda) = \begin{cases} 
\nu(\Lambda), & i = s \\
0, & i \neq s,t \\
-\nu(\Lambda), & i = t
\end{cases}
\]

(9)

\[
\ell(i,j;\Lambda) \leq f(i,j;\Lambda) \leq u(i,j), \quad \forall (i,j) \in A.
\]  

(10)

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 \( f(\Lambda) = (f(i,j;\Lambda))_{(i,j) \in A} \) 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 \( \Lambda \). The parametric residual flow \( \hat{r}(i,j;\Lambda) \) of any arc \((i,j) \in A\), with respect to a given flow \( f(\Lambda) \), is given by:

\[
\hat{r}(i,j;\Lambda) = u(j,i) - f(j,i;\Lambda) + f(i,j;\Lambda) - \ell(i,j;\Lambda)
\]  

(11)

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:

\[
 f(i, j; \lambda) = \hat{r}(i, j; \lambda) - u(i, j) + \ell(i, j; \lambda), 0 \].
\] (12)

**Definition 1**: A parametric cut partitioning \([ S_i, T_i ] \), \( k = 1, ..., K \) together with a partitioning \( J_k \) of the interval of the parameter \([0, \Lambda] \) into disjoints subintervals so that \( J_1 \cup \cdots \cup J_k = [0, \Lambda] \) and \( J_p \cap J_q = \phi \), \( \forall p \neq q \).

The capacity of a parametric \( s-t \) cut partitioning for the minimum flow problem is a piecewise linear function \( \hat{c}[S_i; T_i] \) defined for all \( \lambda \) of every subinterval \( \lambda \in J_k \), \( k = 1, ..., K \):

\[
\hat{c}[S_i; T_i] = \sum_{(i, j) \in \Lambda} \ell(i, j; \lambda) - \sum_{(i, j) \in \Lambda} u(i, j).
\] (13)

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 \([ \hat{S}_i; \hat{T}_i ] \) for all \( \lambda \in \hat{J}_k \) is referred to as a parametric maximum \( s-t \) cut, \([ \hat{S}_i; \hat{T}_i ] \) for the whole interval of the parameter values, \([0, \Lambda] \). Thus a parametric maximum cut \([ \hat{S}_i; \hat{T}_i ] \) is a set of maximum cuts \([ \hat{S}_i; \hat{T}_i ] \) and \( \hat{c}[\hat{S}_i; \hat{T}_i] = \hat{c}[\hat{S}_i; \hat{T}_i] \) for all \( \lambda \) of every subinterval \( \hat{J}_k \), \( k = 1, ..., K \).

**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 \( \lambda^* \), the value of the non-parametric minimum flow with respect to fixed lower bounds \( \ell(i, j; \lambda^*) \), \( f(\lambda^*) = (\ell(i, j; \lambda^*) \cdot \lambda^*, f(\lambda^*) \) equals the capacity of the parametric maximum \( s-t \) cut:

\[
\hat{v}(\lambda^*) = \hat{c}[\hat{S}_i; \hat{T}_i].
\]

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 \([ \hat{S}_i; \hat{T}_i ] \) remains a maximum cut \([ \hat{S}_i; \hat{T}_i ] \) for the entire subinterval \( \hat{J}_k \).

For every subinterval \( \hat{J}_k \) holds that \( \hat{v}(\lambda) = \hat{c}[\hat{S}_i; \hat{T}_i] \), \( \forall \lambda \in \hat{J}_k \) and since \( \hat{c}[\hat{S}_i; \hat{T}_i] = \hat{c}[\hat{S}_i; \hat{T}_i] \) for all \( \lambda \) of every subinterval \( \hat{J}_k \), \( k = 1, ..., K \) 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: \( \hat{v}(\lambda) = \hat{c}[\hat{S}_i; \hat{T}_i] \), \( k = 1, ..., K \) for all \( \lambda \in [0, \Lambda] \).

The sets:

\[
\hat{I}(i, j) = \{ \lambda | \hat{r}(i, j; \lambda) > 0 \} \quad \forall (i, j) \in \hat{A}
\] (14)

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 2**: 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:

\[
\hat{I}(\hat{P}) = \bigcup_{(i, j) \in \hat{P}} \hat{I}(i, j) \neq \phi.
\]

**Definition 3**: 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 of all arcs composing the conditional decreasing directed path for all \( \lambda \in \hat{I}(\hat{P}) \):

\[
\hat{r}(\lambda) = \min_{\lambda \in \hat{I}(\hat{P})} \{ \hat{r}(i, j; \lambda) \} \forall (i, j) \in \hat{P}(\lambda).
\] (15)

**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
4 Minimum flow balancing algorithm for a parametric bipartite network

The balancing algorithm for the minimum flow in a parametric bipartite network proposed in this paper decreases the flow over simple decreasing directed paths in a special parametric bipartite network called monotone parametric bipartite network. The lower bounds of the arcs out of the source are non-increasing functions of a parameter $\lambda$ and the lower bounds of the arcs into the sink non-decreasing functions of $\lambda$ while the lower bounds of the remaining arcs being constant. The parametric lower bound linear functions of $\lambda$ take values in the interval $[0, \Lambda]$, that is:

$$\ell(s,i;\lambda) = \ell_0(s,i) - \lambda \cdot E(s,i)$$  \hspace{1cm} (16)$$

with $0 \leq E(s,i) \leq \ell_0(s,i) / \Lambda \quad \forall (s,i) \in A$

$$\ell(t,j;\lambda) = \ell_0(t,j) - \lambda \cdot E(t,j)$$  \hspace{1cm} (17)$$

with $(\ell_0(t,j) - u(t,j)) / \Lambda \leq E(t,j) \leq 0. \quad \forall (j,t) \in A$.

For the parametric minimum flow problem a flow is referred to as $\lambda$-balanced if there is no simple decreasing directed path from the source node to the sink node in the parametric residual network.

4.1 Balancing algorithm

Let a step $k$ of flow decreasing be considered starting with a feasible flow $f_{k}(\lambda)$. For every arc $(s,i)$ the value $\lambda^k_s$ is defined to be the maximum value of $\lambda$ for which $f_{k}(s,i;\lambda) = (s,i;\lambda)$, i.e. $\lambda^k_s := \max \{ \lambda | \hat{f}(s,i;\lambda) = 0 \}$ and equivalently, for every arc $(j,t)$, the value $\lambda^k_j$ is defined to be the minimum value of $\lambda$ such that $f_{k}(j,t;\lambda) = (j,t;\lambda)$, i.e. $\lambda^k_j := \min \{ \lambda | \hat{f}(j,t;\lambda) = 0 \}$. Actually for a certain value $\lambda^k$ the flow can be decreased along the arc $(s,i)$ only for the parameter values $\lambda \in (\lambda^k, \Lambda]$ since the lower bounds of the arcs out of the source are non-increasing functions of $\lambda$. Similarly, for a value $\lambda^k$ defined for an arc entering the sink the flow can be decreased along the arc $(j,t)$ only for the parameter values $\lambda \in [0, \lambda^k)$ since the lower bounds of the arcs into the sink are non-decreasing functions of $\lambda$. Consequently the flow can be decreased along a conditional decreasing directed path including the arcs $(s,i)$ and $(j,t)$ as long as $[0, \lambda^k) \cap (\lambda, \Lambda] \neq \phi$, 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, $\lambda^k_i < \lambda^k_j$. According to the general definition, a flow $f(\lambda)$ is said to be $\lambda$-balanced if there is no decreasing path from a node $i$ to a node $j$ with $\lambda_i < \lambda_j$. For the monotone parametric bipartite network in discussion, a flow is referred to as $\lambda$-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 $\phi$ if not constrained by the residual capacity of the arc $(i,j)$.

4.2 Decreasing the flow along simple directed paths in residual network

Definition 4: A directed path $\hat{P} = (s,i,j,t)$ with $i \in N_s - \{ t \}$ and $j \in N_i - \{ s \}$ in $\hat{C}(\lambda,f)$=$(N,\hat{A})$ is referred to as a simple residual path if one of the following two cases holds:

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

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

Let a step of flow decreasing be considered over a simple residual path, $\hat{P} = (s,i,j,t)$, starting with an actual feasible flow $f(\lambda)$ and let $\lambda^k_i$ and $\lambda^k_j$ denote the lambda values at the beginning of a decreasing step respectively $\lambda^{k+1}_i$, $\lambda^{k+1}_j$ the corresponding values at the end of the decreasing step. For the case when $\lambda_i < \lambda_j$, 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 \(\phi\), i.e.
\[\lambda_{k+1}^i = \lambda_{j+1}^i = \lambda^i.\]
The maximum value \(\hat{r}(\hat{P}, \lambda^i)\) of the piecewise linear residual capacity function for which the two new lambda values become equal after the pull is computed as \(\hat{r}(\hat{P}, \lambda^i) = \hat{r}(s, i; \lambda^i) = \hat{r}(j, t; \lambda^i)\) if not constrained by residual capacity of the arc \((i,j)\).

Otherwise, the decreasing of flow will make the two new lambda values \(\lambda_{k+1}^i, \lambda_{j+1}^i\) come closer to each other, their new values being computed as solutions of the following equations:
\[\hat{r}(s, i; \lambda_{k+1}^i) = \hat{r}(i, j)\]
and
\[\hat{r}(j, t; \lambda_{j+1}^i) = \hat{r}(i, j).\]
For this case, after the pull of flow, the residual capacity of the arc \((i,j)\) will become zero for all the values of the parameter: \(\lambda \in (\lambda_{k+1}^i, \lambda_{j+1}^i).\)

Hence, a pull of \(\hat{r}(\hat{P}, \lambda)\) flow is performed from the sink node to the source node along the directed path \(\hat{P}(\lambda)\) in \(\hat{G}(\lambda, f)\). For the second case, when \(\lambda^i > \lambda^j\), a pull of flow equal to \(\hat{r}(\hat{P}, \lambda)\) is performed from the source node \(s\) to the sink node \(t\) through the reversed directed path \(\hat{P}(\lambda)\) in \(\hat{G}(\lambda, f)\). If the residual capacity of the arc \((j,i)\) restricts the pull of flow by being smaller then the maximum value \(\hat{r}(\hat{P}, \lambda^i)\), i.e.
\[\hat{r}(j, i; \lambda) < \hat{r}(\hat{P}, \lambda^i)\]
for \(\lambda \in (\lambda^i, \lambda^j)\), then the two new lambda values \(\lambda_{k+1}^i, \lambda_{j+1}^j\) come closer to each other and are computed as solutions of the following equations:
\[\ell(s, i; \lambda_{k+1}^i) = \ell(s, i; \lambda^i) + \hat{r}(j, i)\]
and respectively
\[\ell(j, t; \lambda_{j+1}^j) = \ell(j, t; \lambda^j) + \hat{r}(i, j).\]
Thus:

(a) If \(\lambda^i < \lambda^j\), 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 \(\hat{G}(\lambda, f)\) is performed;

(b) If \(\lambda^j > \lambda^i\), a pull of \(\hat{r}(\hat{P}, \lambda)\) flow from the source node \(s\) to the sink node \(t\) through the reversed directed path \(\hat{P}(\lambda)\) in \(\hat{G}(\lambda, f)\) is performed.

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

(a) if \(\lambda^i < \lambda^j\), then \(\lambda^i < \lambda_{k+1}^i \leq \lambda_{j+1}^j < \lambda^j\)

(b) if \(\lambda^j > \lambda^i\), then \(\lambda^i > \lambda_{j+1}^j \geq \lambda_{k+1}^i > \lambda^j\).

After each step of the decreasing 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 it could be reversed by other operations.

### 4.3 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 \(G = (N, A, c, s, t)\) obtained from the initial network \(G(\lambda) = (N, A, \ell, u, s, t)\) by modifying the parametric lower bounds as follows:
\[\ell^*(s, i) = \left\{ \begin{array}{ll}
\ell(s, i) & \text{if } \ell(s, i) > \ell^*(i, j) - \lambda \\
\ell(i, j) & \text{otherwise}
\end{array} \right.\]

In the second phase the algorithm maintains a set \(L\) of nodes \(j \in N_s - \{s\}\) whose last examination resulted in a change in the flow. When \(L\) becomes empty it is reset to \(L := N_s - \{s\}\). If an entire pass over \(L\) results in no flow change the computation is complete and a parametric minimum flow \(\hat{f}(\lambda)\) is obtained.

**Programme BMPB:**

1. **Begin**
2. find an initial feasible flow \(f\) in \(G\);
3. compute \(\hat{G}(\lambda, f)\);
4. **for** \(i := 1\) **to** \(n\) **do** \(\lambda_i := 0\);
5. **for** \(j := 1\) **to** \(m\) **do** \(\lambda_j := \lambda\);
6. \(L_0 := \emptyset\);
7. **for** \(j := 1\) **to** \(n\) **do** add node \(j\) to \(L_0\);
8. \(L := L_0\);
9. **while** \(L \neq \emptyset\) **do**
10. \(B := 0\);
11. remove the first node \(j\) from \(L\);
12. select the first arc \((i, j)\) with \(i \neq t\);
13. **repeat**
14. **if** \((\hat{r}(i, j) > 0)\) and \((\lambda < \lambda_j)\) **then**
15. **begin**
16. pull \(\hat{f}(\hat{P}, \lambda)\) through \(\hat{P}(\lambda) = (s, i, j, t)\);
17. \(B := 1\);
18. **end**
19. **else** **if** \((\hat{r}(j, i) > 0)\) and \((\lambda > \lambda_i)\) **then**
20. **begin**
21. pull \(\hat{f}(\hat{P}, \lambda)\) through \(\hat{P}(\lambda) = (t, j, i, s)\);
22. \(B := 1\);
23. **end**
24. **if** \(B = 0\) **then** update \(\lambda_i, \lambda_j\) and \(\hat{G}(\lambda, f)\);
25. **if** \((i, j)\) is not the last arc adjacent to \(j\) **then**
26. **begin**
27. select the next arc \((i, j)\) with \(i \neq t\);
28. \(C := 1\);
29. **end**
30. **until** \((C = 0)\);
31. **if** \((B = 1)\) **then** add node \(j\) to \(L\);
32. **if** \((L = 0)\) and \((B = 1)\) **then** \(L := L_0\);
33. **end**
34. **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 \( \lambda_i \in (0, \Lambda] \) the nodes in \( N - \{s, t\} \) are partitioned according to their lambda values in two disjoint sets \( S_i \) and \( T_i \). Let \( S_i \) be the set of nodes \( i \in N - \{s, t\} \) with \( \lambda_i < \lambda_j \), \( S_i \) := \{i | \lambda_i < \lambda_i\} and \( T_i := \{j | \lambda_j \geq \lambda_i\} \) which means that the two sets generates the \( s - t \) cut \( [S_i, T_i] \). For every arc \( (i, j) \in (S_i, T_i) \) 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 \( (j, i) \in (T_i, S_i) \) the residual capacity of the arc \( \hat{r}(i, j; \lambda) = 0 \) 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 \( [S_i, T_i] \) 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 \( \hat{r}(\hat{P}, \lambda) = 0 \). Hence for all lambda values the flow \( f(\lambda) = f(i, j; \lambda) \) is a parametric minimum flow.

4.4 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 \( \lambda \)-values close enough that a simple post processing phase will complete the computation.

Theorem 6: The \( \varepsilon \)-approximation of the BMPB has the complexity of \( O(n_1 \cdot n_2 \log(\lambda/\varepsilon)) \).

Proof: Let the flow be decreased over an arc \( (i, j) \) with positive residual capacity only if \( \lambda_i + \varepsilon < \lambda_j \) for a desired \( \varepsilon \)-approximation. Pulling flow over all arcs \( (i, j) \) entering in the same node \( j \) results in each of the iterations in \( \lambda_i \)-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 \( \lambda \)-values and the minimum difference between \( \lambda \)-values, e.g. \( R := \Lambda/\varepsilon \). The algorithm stops after \( O(\log(\lambda/\varepsilon)) \) passes over the arcs taking \( O(n_1 \cdot n_2) \) time, for a total time of \( O(n \cdot n_2 \log(\lambda/\varepsilon)) \).

5 Conclusion

Even if the type of parameterization in this paper appears to be quite specialized, Gallo et. al. [4] 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. Although this algorithm is not competitive with other algorithms, it has the advantage of being extremely simple and intuitive.

References:


