A GREEDY ALGORITHM FOR MULTICUT AND INTEGRAL MULTIFLOW IN ROOTED TREES

MARIE-CHRISTINE COSTA [1], LUCAS LÉTOCART [1] AND FRÉDÉRIC ROUPIN [2]

(1) *CEDRIC, CNAM, 292 rue St-Martin 75141 Paris cedex 03, France.*

(2) *CEDRIC, CNAM-IIE, 18 allée Jean Rostand 91025 Evry cedex, France.*

e-mails : {costa,letocart}@cnam.fr, roupin@iie.cnam.fr.

Abstract. We present an $O\left(min\left(Kn, n^2\right)\right)$ algorithm to solve the maximum integral multiflow and minimum multicut problems in rooted trees, where K is the number of commodities and n is the number of vertices. These problems are NP-hard in undirected trees but polynomial in directed trees. In the algorithm we propose, we first use a greedy procedure to build the multiflow then we use duality properties to obtain the multicut and prove the optimality.

keywords. maximum integral multiflow, minimum multicut, duality, rooted tree.

1. Introduction

Consider a graph $G = (V, E)$ with a positive capacity (or weight) on each edge of E and a list of K pairs of vertices $\{s_k, t_k\}$, $k \in \{1, .., K\}$. Associate a commodity with each vertex pair $\{s_k, t_k\}$. The integral multiflow problem consists in maximizing the sum over all commodities of the integral flow corresponding to a commodity subject to capacity and flow conservation requirements. The multicut problem is to find a minimum weight set of edges whose removal separates each pair $\{s_k, t_k\}$ of the list. For $K = 1$ the problems are the ordinary min cut-max flow problem solvable in polynomial time but both problems are known to be NP-hard and even MAX SNP-hard for $K \geq 3$ [2], [3]. Garg, Vazirani and Yannakakis [4] proved that these complexity results hold even if the graph is a tree. However, they give a polynomial algorithm for the integral multiflow in trees with unit capacities on the edges. Nevertheless, these problems are polynomial for any directed tree *i.e.* for any tree in which edges can be oriented to obtain a (unique) directed path from s_k to t_k for each k. In fact, the multiflow problem in a di-tree can be reduced to a minimum cost circulation problem in a directed graph obtained by adding to the tree the edges from t_k to s_k for all $k \in \{1, .., K\}$: costs are set to -1 on the added edges and set to 0 on the original edges. Polynomial algorithms were proposed for the circulation problem [5] and can be used for the multiflow problem in a tree; the most efficient was proposed by Orlin [6] and is in $O\left((m + nlogn)\, mlogn\right)$ where m is

the number of edges and n is the number of vertices. The number of edges of the derived di-graph is $n+K-1$ thus the overall complexity of Orlin's algorithm is here $O\left(max\left(K^2logn,\, n^2log^2n\right)\right)$. In this paper, we propose a greedy algorithm in $O\left(min\left(Kn, n^2\right)\right)$ to solve the integral multiflow and multicut problems in a rooted tree, *i.e.* a directed tree admitting a vertex, the root, from which there is a path to any other vertex.

The paper is organized as follows. In the next section we give the mathematical models of the problems and we recall the dual relationship of their continuous relaxations [3], [7]. In section 3 we use this duality to build the greedy algorithm in a rooted tree and prove its correctness. Next we evaluate its complexity before concluding.

2. Two continuous dual problems

Shmoys [7] presents the dual relationship between the integral multiflow and multicut problems in general graphs (see also [3]). Let us give the models when the graph is a tree $T=(V,E)$. The models and all the results hereafter are valid if T is a rooted tree. For each edge e of E, denote by u_e the capacity of the edge e (u_e is assumed to be positive and integral). Let p_k be the path from s_k to t_k and let f_k, $k \in \{1,..,K\}$ and c_e, $e \in E$, be the variables: f_k is the flow on the path p_k and $c_e = 1$ if the edge e belongs to the cut, $c_e = 0$ otherwise. The integral multiflow problem $(IMFP)$ and the $s_k - t_k$ multicut problem $(IMCP)$ can be formulated as:

$$
\begin{array}{|lll|}
\hline
(IMFP) & & \\
\text{Max} & \sum_{k=1}^{K} f_k & \\
\text{s. t.} & \sum_{k\ st\ e\in p_k} f_k \ \leq u_e & \forall e \in E \quad (1) \\
& f_k \in \mathbb{N} & \forall k \in \{1,..,K\} \\
\hline
\end{array}
\qquad
\begin{array}{|lll|}
\hline
(IMCP) & & \\
\text{Min} & \sum_{e\in E} u_e\, c_e & \\
\text{s. t.} & \sum_{e\in p_k} c_e \ \geq 1 & \forall k \in \{1,..,K\} \quad (2) \\
& c_e \in \{0,1\} & \forall e \in E \\
\hline
\end{array}
$$

Denote by $(CMFP)$ and $(CMCP)$ the continuous relaxations obtained from $(IMFP)$ and $(IMCP)$ by replacing the integrality constraints by nonnegativity constraints (note that constraints $c_e \leq 1\ \forall e \in E$ are useless). Let f^* and c^* be optimal solutions of $(CMFP)$ and $(CMCP)$. $(CMFP)$ and $(CMCP)$ are two dual linear programs and the complementary slackness conditions of optimality in linear programming are given by:

$$\forall k \in \{1,..,K\} \quad f_k^* > 0 \ \Rightarrow \sum_{e\in p_k} c_e^* = 1 \qquad (3)$$

$$\forall e \in E \qquad c_e^* > 0 \ \Rightarrow \sum_{k\ s.t.\ e\in p_k} f_k^* \ = u_e \quad (4)$$

The constraints (3) imply that if the variables c_e^* are integral then they define the incident vector of a multicut C, and there is exactly one edge of p_k in C for all k such that $f_k^* > 0$. The

constraints (4) imply that if the variables f_k^* and c_e^* are integral then all the edges in the associated multicut C are saturated edges, *i.e.* edges with residual capacities equal to zero.

If the graph is a directed tree, the constraint matrix in the program $(IMFP)$ is a submatrix of a Chain matrix. Recall that a Chain matrix is a matrix whose columns are all the edge vectors of directed paths in a graph. P. Camion [1] showed that the Chain matrix defined in a directed tree is totally unimodular. Therefore the constraint matrix of the multiflow is totally unimodular and the constraint matrix of the dual multicut program $(IMCP)$ is totally unimodular too. The integral multiflow and multicut problems in directed trees can be solved by linear programming and so are polynomial. This is also a consequence of the reduction to a minimum cost circulation problem in a directed graph as shown in the introduction. In the next section, we present a greedy algorithm to solve both problems in rooted trees. Unfortunately, we shall see that it cannot be applied to a directed tree which does not admit a root.

3. A GREEDY ALGORITHM IN ROOTED TREES

We assume in this section that the graph is a rooted tree and we propose to solve first the integral multiflow problem (subsection 3.1) and then the multicut problem (subsection 3.2). The basic idea of the algorithm is to find integral solutions verifying the complementary slackness conditions. The proof of correctness and the complexity evaluation of our algorithm are given respectively in subsections 3.3 and 3.4.

3.1. **The integral multicommodity flow procedure.** The procedure begins with the leaves of the rooted tree and then covers the nodes level by level up to the root, widthwise. Each time a source s_k is encountered a maximal quantity of flow is routed on p_k, saturating at least one new edge if $f_k > 0$. If there is more than one source in a node, any order can be considered for these sources. See procedure $Maxmultiflow$ hereafter and figure 1 for an example. The procedure is valid for any value of K. However, to improve its complexity in the case $K = O(n^2)$ we need to slightly modify it: the changes are indicated in square brackets in the procedure.

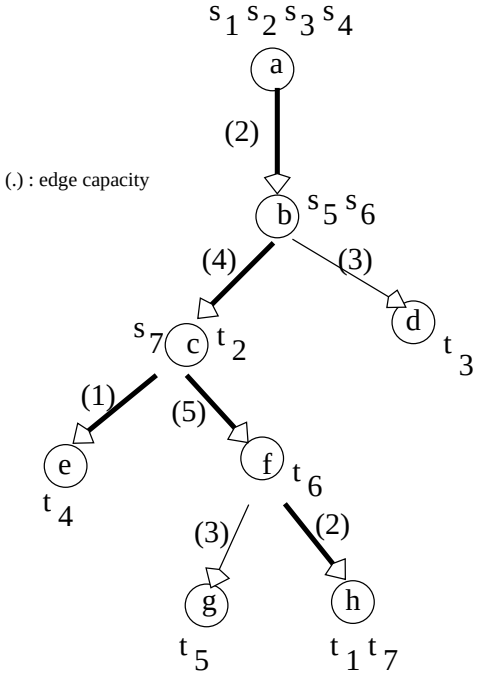
The edge set $C_0 \subseteq E$ will contain edges saturated all along the procedure. If the capacities of the edges are integers, then all the routed flows f_k are integers. At the end of the procedure the value of the integral multiflow $\widehat{F}$ is equal to $\Phi = \sum_{k=1}^{K} f_k^*$ and there is at least one saturated edge (in C_0) on each path p_k.

3.2. **The multicut procedure.** The multicut is built from the multiflow solution obtained in subsection 3.1. The procedure considers the sources in the order given by the numbering obtained

procedure $Maxmultiflow$;
input : $T=(V,E)$, let $a,b,c,...$ be a breadth-first lexicographic ordering on the vertices of V (obtained in $O(n)$)
$(s_k,\ t_k) \in V^2 \quad \forall k \in \{1,...,K\}$.
output : $(f_1^*,...,f_k^*,...,f_K^*)$ a maximal multiflow. C_0 the set of edges saturated by the multiflow
1. Number the flows considering the sources from the root down to the leaves widthwise;
2. $C_0 \leftarrow \emptyset$; $\quad$ [*if* $K=O(n^2)$ *then add :* $L \leftarrow list\ \{f_K,\ f_{K-1},...,\ f_1\}$]
3. **for** $k=K$ **to** 1 **do** $\quad$ [*if* $K=O(n^2)$ *then replace by* **while** $L \neq \emptyset$ **do**]
[*if* $K=O(n^2)$ *then add* $k \leftarrow$index of the first flow of L]
//from the leaves up to the root
route the maximal flow from s_k to t_k, f_k^*, with respect to the current residual capacities;
$E_k \leftarrow$*{new edges saturated by* f_k^**}*; $\quad$ [*if* $K=O(n^2)$ *then add :*$L \leftarrow L-\{f_k\}$;]
$C_0 \leftarrow C_0 \bigcup E_k$; *//there is at least one more edge in* C_0 *if* $f_k^*>0$
[*if* $K=O(n^2)$ *then add :* **for** f_i such that $p_i \cap E_k \neq \emptyset$ **do** $f_i^* \leftarrow 0$; $L \leftarrow L-\{f_i\}$ **end do;**
//$\exists$ *a saturated edge in* p_i]
end do;
end $Maxmultiflow$;

by $Maxmultiflow$, from the root down to the leaves. In order to satisfy the complementary slackness conditions (4) all the edges of the cut are selected from C_0 and so are saturated edges. Moreover, in order to satisfy constraints (3), we keep only one edge e on each path p_k such that $f_k^*>0$. At step k, e is either the only or the "highest" remaining saturated edge on p_k, the highest saturated edge being the first saturated edge encountered on the path from the source s_k to the sink t_k; then we definitively remove all the other saturated edges belonging to p_k. Doing so we get a set $\widehat{C} \subseteq E$ which contains at most one edge on each path p_k such that $f_k^*>0$. See procedure $Minmulticut$ and figure 1 for an example. Let Γ be the value of the set $\widehat{C}$, $\Gamma=\sum_{e\in\widehat{C}} u_e$. In the next section we shall prove that $\widehat{C}$ is a multicut and that $\Gamma=\Phi$.

procedure $Minmulticut$
input : $T=(V,E)$, $s_k-t_k \quad k\in\{1,...,K\}$, f^*, C_0=*{edges saturated by the multiflow}*
output : a minimal multicut $\widehat{C}$.
for $k=1$ **to** K **do** *//numbering given by Maxmultiflow*
//paths and flows are considered from the root down to the leaves
if $f_k^*>0$ **and** $|C_{k-1} \bigcap p_k|>1$ **then**
//only one edge of $\widehat{C}$ *allowed on* p_k
$e_k \leftarrow first\ saturated\ edge\ on\ the\ path\ from\ s_k\ to\ t_k$;
$C_k \leftarrow [C_{k-1}-(C_{k-1} \bigcap p_k)] \bigcup \{e_k\}$;
//$C_{k-1} \bigcap p_k$ *is a set of edges with a residual capacity equal to zero*
//suppress all the edges of p_k *in* C_{k-1} *except* e_k
else
//either $f_k^*>0$ *and there is at most one remaining edge of* p_k *in* C_{k-1}
//or $f_k^*=0$ *and more than one edge of* $\widehat{C}$ *are allowed on* p_k*. Nothing to do*
$C_k \leftarrow C_{k-1}$;
endif;
end do;
$\widehat{C} \leftarrow C_K$;
end $Minmulticut$;



Computational order for the flow:

f7=2, f6=3, f5=0, f4=1, f3=1, f2=0, f1=0

Computational order for the cut:

C0={(a,b),(b,c),(c,e),(c,f),(f,h)}

C0=C1=C2=C3

C4=C3-{(b,c),(c,e)}={(a,b),(c,f),(f,h)}

C4=C5=C6

C7=C6-{(f,h)}={(a,b),(c,f)}

Fopt=(0,0,1,1,0,3,2) Copt={(a,b),(c,f)}

Γ=Φ=7

FIGURE 1. Application of the algorithm in a rooted tree

3.3. **Optimality of the algorithm.** To prove the optimality of the algorithm, we first show that the cut $\widehat{C}$ is indeed a multicut whose removal separates each pair $\{s_k, t_k\}$, $k \in \{1, .., K\}$, and then that there is no duality gap between the multiflow and the multicut values.

Lemma 1. *The set $\widehat{C} \subseteq E$ obtained at the end of the algorithm contains at least one edge on each path p_k from s_k to t_k, $k \in \{1, .., K\}$: $\widehat{C}$ is a multicut.*

Proof. The proof is obtained by induction on i. From the principle of the $Maxmultiflow$ procedure, we know that there is at least one edge of each p_k, $k \in \{1, .., K\}$, in the set C_0. We suppose that this property holds for the set C_{i-1} $(i > 0)$.

Let us show that the property holds for C_i.

If $f_i^* = 0$ then $C_i = C_{i-1}$ and the property holds for C_i. The next step is to study $C_i \bigcap p_k$ for i such that $f_i^* > 0$, and for all the paths p_k, $k \in \{1, .., K\}$.

There is at least one edge of p_i in C_{i-1} and at step i in the $Minmulticut$ procedure we keep one of these edges, so the property is true for $k = i$.

Consider now the paths p_k, $k \in \{1, .., K\}$, $k \neq i$: they belong to one of the three classes hereafter:

(a) $p_k \bigcap p_i = \emptyset$. There was an edge of p_k in C_{i-1} and at step i we only suppress edges of p_i to get C_i; so there is an edge of p_k in C_i too.

(b) $p_k \bigcap p_i \neq \emptyset$ and $k < i$. Then $C_i \subseteq C_{i-1} \subseteq C_k$. By the hypothesis, there exists an edge e_k of p_k in C_{i-1} and by *Minmulticut*, e_k is the only edge of p_k in C_k, so e_k is the only edge of p_k in C_{i-1}. Moreover, e_k is not suppressed by *Minmulticut* at step i because either $e_k \notin p_i$ or e_k is the first saturated edge on the path from s_i to t_i (because e_k is the first saturated edge on p_k, $e_k \in p_i$ and s_i is on p_k) and so $e_k = e_i$ (recall that for j in $\{1, ..., K\}$, e_j is the first saturated edge on the path p_j).

(c) $p_k \bigcap p_i \neq \emptyset$ and $k > i$. Then, $C_{k-1} \subseteq C_i$. A saturated edge v exists in p_k after f_k being routed and before routing f_i. For all j such that $j < k$ and $f_j^* > 0$ we have $v \notin p_j$ (otherwise f_j being routed after f_k would be equal to 0). $v \in C_0$ and v cannot be suppressed at any step $j < k$ in *Minmulticut*; so $v \in C_{k-1}$ and then $v \in C_i$.

We conclude that for all $k \in \{1, .., K\}$ there is an edge of p_k in C_i: C_i is a multicut. □

Lemma 2. *Let Φ and Γ be the values of the integral multiflow $\widehat{F}$ and of the multicut $\widehat{C}$ obtained at the end of the algorithm. Then $\Phi = \Gamma$, $\widehat{F}$ is a maximum integral multicommodity flow and $\widehat{C}$ is a minimum multicut.*

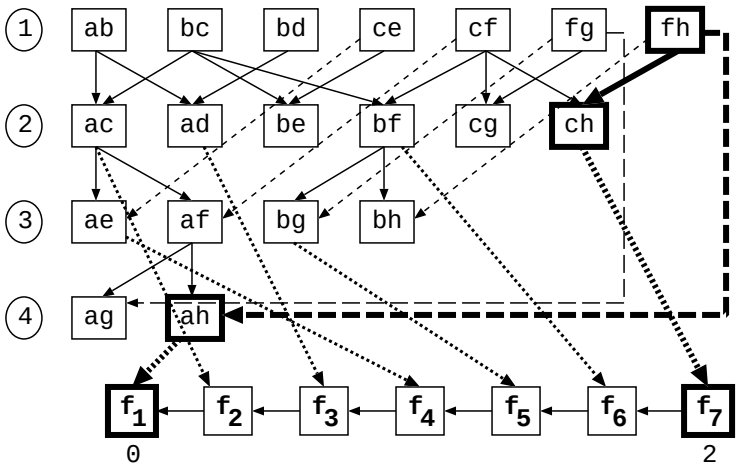
Proof. At the end of the algorithm, $\widehat{C}$ contains one and only one edge on each path p_k such that $f_k^* > 0$. The solution of the program $(IMCP)$ is given by: $c_e^* = 1$ if $e \in \widehat{C}$; $c_e^* = 0$ otherwise; for all $e \in E$. The complementary slackness conditions are verified: if $f_k^* > 0$ for some k then, from the *Minmulticut* and the *Maxmultiflow* procedures we know that there is one and only one edge of p_k in $\widehat{C}$ so $\sum_{e \in p_k} c_e^* = 1$; moreover, all edges in $\widehat{C}$ belong to C_0 and are saturated edges, so if $c_e^* > 0$ for some e then $\sum_{k \ s.t. \ e \in p_k} f_k^* = u_e$. The solutions of $(IMFP)$ and $(IMCP)$ associated with $\widehat{F}$ and $\widehat{C}$ satisfy the complementary slackness conditions of linear programming, so they are optimal and $\Phi = \Gamma$. □

3.4. **Complexity of the algorithm.** First, it is easy to verify that *Minmulticut* can be implemented within an $O\left(min\left(Kn, n^2\right)\right)$ complexity. Indeed, for all $k \in \{1, ..., K\}$, $|C_k| \leq n$ and the path p_k has less than n edges. There are at most $min(K, n)$ positive flows f_k^* and the computation of e_k and C_k needs $O(n)$ time. The **else** part is in $O(K)$. The overall complexity is $O\left(n\, min\left(K, n\right) + K\right)$ *i.e.* $O\left(min\left(Kn, n^2\right)\right)$.

Second, we claim that the procedure *Maxmultiflow* can be implemented also within an $O\left(min\left(Kn, n^2\right)\right)$ complexity. First, at step 1, each source is seen once, so this step can be done in $O\left(max\left(K, n\right)\right)$ time. Second, let us consider step 3. Routing each flow f_k takes at most $O\left(n\right)$ time in a tree. Now we study two different cases.

If $K = O(n)$ then step 3 takes $O(Kn)$ time.

If $K = O(n^2)$ we need a special datastructure.



f_7 is routed ($f_7^* = 2$) and removed from L, the edge fh is saturated;
fh, ch, ah and f_1 are removed from the datastructure, f_1^* is set to 0.

FIGURE 2. $K = O(n^2)$: datastructure associated to figure 1

We use a linked-list L of the flows ordered from f_K to f_1 (last line of figure 2) and a list of edges of E ordered according to the lexicographic order (first line of figure 2). In addition, we build all paths of T into our datastructure: in level j of the datastructure we list all paths of length j, defining a pointer from an edge e at level 1 and a path p at level j to a path q at level $j+1$ (if q is obtained from p by adding e at the end of it). We also define a pointer from each path to the corresponding flow in the last level. Thus we have at most two pointers to each path at each level $j \geq 2$. The pointers describe inclusions of paths (not all inclusions, but enough to ensure that all paths containing an edge e are reachable from e in our datastructure). Indeed, there are several ways to obtain a path of length j ($j \geq 2$): we have chosen to add an edge at the end of a path of length $j-1$. For example, the path ah of figure 1 is obtained from af and fh (and not from ab and bh).

The vertices being labelled following a breadth-first lexicographic order, to build the list of level j we need to traverse once the list $j-1$ and at the same time once the list of level 1. The list of level 1 is ordered in a lexicographical way and thus the lists of level $2, ..., j$ are naturally obtained in the lexicographic order in $O(n)$.

There are at most $n-1$ paths of length j for any j in $\{1, ..., n-1\}$, the total number of paths is thus at most n^2, and the total number of pointers is less than $2n^2$. We set a pointer from each

path to the corresponding flow in L if it exists (K pointers). Finally, the total number of pointers is less than $2n^2 + K$, and the overall complexity to build our data strucure is $O\left(n^2\right)$.

At each step of loop 3 of our procedure $Maxmultiflow$, first we route a positive flow f_k and remove it from L. Second we consider the new saturated edges. Third, thanks to our datastructure, we set to zero and remove from the datastructure all flows corresponding to paths containing a new saturated edge. There are at most n positive flows to route. Each routing takes $O\left(n\right)$ time and implies the removal of a part of the datastructure. Hence, the total routing time is in $O\left(n^2\right)$ and the datastructure is traversed once to remove its $O\left(n^2\right)$ items: the whole complexity of $Maxmultiflow$ when $K = O\left(n^2\right)$ is $O\left(n^2\right)$.

Finally, the procedure $Maxmultiflow$ is in $O\left(min\left(Kn,\, n^2\right)\right)$ as the procedure $Minmulticut$. This means that the global complexity is $O\left(min\left(Kn,\, n^2\right)\right)$.

4. Conclusion

We have proposed an $O\left(min\left(Kn, n^2\right)\right)$ algorithm for integral multicommodity flow and multicut problems in rooted trees. Unfortunately, the greedy procedure for rooted trees defined in this paper and adapted to general ditrees does not work. Just consider the simple counter-example obtained by slightly modifying the example given in Figure 1: we add to the tree an edge $(a'b)$ with weight $u_{a'b} = 1$ and a flow f_0 with $s_0 = a'$ and $t_0 = e$. The solution obtained by the algorithm is the solution of Figure 1 with $f_0 = 0$ and $\Phi = 7$ whereas the optimal solution is $F^* = (1, 0, 0, 2, 0, 0, 3, 2)$ with $\Phi^* = 8$. We have tried many different numberings of the commodities unsuccessfully: for each one we have found a counter-example.

Using Orlin's algorithm to solve max multiflow in di-trees (see introduction for details), one gets an overall complexity of $O\left(n^2 log^2 n\right)$ when $K = O\left(n\right)$ and $O\left(n^4 logn\right)$ when $K = O\left(n^2\right)$. Our algorithm is better than Orlin's one regarding the complexity (in rooted trees), which is not surprising since Orlin's algorithm can be applied on a more general problem. Moreover, our algorithm provides an optimal multicut.

We thank the referee for her numerous and constructive remarks, especially about the similarity to the circulation problem.

References

[1] P. Camion. 1963. Matrices totalement unimodulaires et problèmes combinatoires. Université de Bruxelles, Thèse et Rapport Euratom.

[2] E. Dalhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. 1994. The complexity of multiterminal cuts. SIAM J. Comput 23, 864-894.

[3] N. Garg, V.V. Vazirani and M. Yannakakis. 1996. Approximate max-flow min-(multi)cut theorems and their applications. SIAM J. Comput. 25, 2. 235-251.

[4] N. Garg, V.V. Vazirani and M. Yannakakis. 1997. Primal-Dual approximation algorithms for integral flow and multicut in trees. Algorithmica 18, 3-20.

[5] A. V. Goldberg, E. Tardos ans R.E. Tarjan. Network flow algorithms. In B. Korte, L. Lovasz, H.J. Prömel and A. Schrijver. 1990. *Paths, Flows and VLSI-Layout.* Algorithms and combinatorics 9. Springer-Verlag. Berlin. 329-371.

[6] J. B. Orlin. 1993. A faster strongly polynomial minimum cost flow algorithm. Operations research. 41,2. 338-349.

[7] D. B. Shmoys. 1997. Cut problems and their application to divide-and-conquer. In D. Hochbaum. *Approximation algorithms for NP-hard problems.* PWS Publishing Company. Boston. 192-234.