

# NETWORK REINFORCEMENT

FRANCISCO BARAHONA

ABSTRACT. We give an algorithm for the following problem: given a graph  $G = (V, E)$  with edge-weights and a nonnegative integer  $k$ , find a minimum cost set of edges that contains  $k$  disjoint spanning trees. This also solves the following *reinforcement problem*: given a network, a number  $k$  and a set of candidate edges, each of them with an associated cost, find a minimum cost set of candidate edges to be added to the network so it contains  $k$  disjoint spanning trees. The number  $k$  is seen as a measure of the invulnerability of a network. We show that this can be solved with  $|V|$  applications of the minimum cut algorithm of Goldberg & Tarjan. The previously known algorithm requires solving  $2|E|$  minimum cut problems.

## 1. INTRODUCTION

Consider a graph  $G = (V, E)$ , for a family of disjoint vertex-sets  $\{S_1, \dots, S_p\}$  let  $\delta(S_1, \dots, S_p)$  be the set of edges with endpoints in different sets of this family. It has been proved in [Tutte, 1961] and [Nash-Williams, 1961] that the maximum number of edge-disjoint spanning trees in  $G$  is

$$(1) \quad \sigma = \min \left\lfloor \frac{|\delta(S_1, \dots, S_p)|}{p-1} \right\rfloor,$$

where the minimum in (1) is taken among all partitions  $\{S_1, \dots, S_p\}$  of  $V$ . The number  $\sigma$  has been proposed as a measure of the invulnerability of a network in [Gusfield, 1983]

In general suppose that each edge  $e$  has a *strength*  $s(e) \geq 0$ . Let  $s(A) = \sum_{e \in A} s(e)$ . The *strength* of this network is

$$(2) \quad \sigma(G, s) = \min \frac{s(\delta(S_1, \dots, S_p))}{p-1}$$

where the minimum in (2) is taken among all partitions  $\{S_1, \dots, S_p\}$  of  $V$ .

Algorithms for computing  $\sigma(G, s)$  have been given in [Cunningham, 1985] and [Cheng and Cunningham, 1994]. In this last reference it is shown that it can be computed in the same asymptotic complexity as  $|V|$  applications of the minimum cut algorithm of [Goldberg and Tarjan, 1988]; this is based on the parametric minimum cut algorithm of [Gallo et al., 1989].

Now suppose that there is a per-unit cost  $c(e)$  of increasing the strength of each edge  $e$ , and a number  $\sigma_0$ . The *reinforcement problem* consists of finding a minimum cost way to increase edge strengths so that the resulting network has strength at least  $\sigma_0$ . An algorithm for this was given in [Cunningham, 1985], it requires solving  $2|E|$  minimum cut problems. In this paper we show that as for the strength problem, it can be solved in the same asymptotic complexity as  $|V|$  applications of the minimum cut algorithm of [Goldberg and Tarjan, 1988]. Although the result sounds similar to the one for the strength problem, the algorithm presented here is quite different.

Now we describe the basics of our approach. Consider a slightly different question, suppose that each edge  $e$  has a non-negative per-unit cost  $d(e)$  and a non-negative integer capacity  $u(e)$ , that gives the maximum number of copies allowed of edge  $e$ . For a non-negative number  $k$ , consider the problem of choosing a minimum cost spanning subgraph of strength  $k$ . This can be modeled as the linear program below.

$$\begin{aligned}
 (3) \quad & \text{minimize } dx \\
 & \text{subject to} \\
 (4) \quad & x(\delta(S_1, \dots, S_p)) \geq k(p-1), \text{ for all partitions } \{S_1, \dots, S_p\} \text{ of } V, \\
 (5) \quad & 0 \leq x(e) \leq u(e).
 \end{aligned}$$

We are going to give a combinatorial algorithm to solve (3)-(5). For the case when  $u(e) = 1$  for all  $e \in E$ , an  $O(|E| \log |E| + k^2 |V|^2)$  algorithm has been given in [Roskind and Tarjan, 1985]. We can reduce the reinforcement problem to this problem by allowing parallel edges and giving the weight zero to the already existing edges.

This paper is organized as follows. In Section 2 we study the question of given a number  $k$  decide if the network has strength at least  $k$ . This is a key subroutine for the reinforcement problem. In Section 3 we describe the minimum cut algorithm of Goldberg & Tarjan and its adaptation to solve a sequence of related minimum cut problems. In Section 4 we give the algorithm for the problem studied in this paper.

We finish this introduction with some notation. Sometimes we shall use the notation  $\delta_G(S_1, \dots, S_p)$  to express the fact that this edge set corresponds to edges in  $G$ . We are going to use  $\delta(S)$  to denote  $\delta(S, V \setminus S)$ , this is called an *undirected cut*.

## 2. TESTING THE STRENGTH

First we study the following question: Given a nonnegative number  $k$ , is the strength of the network greater than or equal to  $k$ ? This reduces to test whether

$$s(\delta(S_1, \dots, S_p)) \geq k(p-1), \text{ for all partitions } \{S_1, \dots, S_p\} \text{ of } V.$$

We shall see later that this is a key subroutine in the solution of (3)-(5).

Dividing the strengths  $s$  by  $k$ , the problem reduces to the following. Given a set of nonnegative weights  $w(e)$  for all  $e \in E$ ,

$$(6) \quad \text{minimize } w(\delta(S_1, \dots, S_p)) - (p-1),$$

where the minimization is done over all partitions  $\{S_1, \dots, S_p\}$  of  $V$ . This has been called the *attack problem* in [Cunningham, 1985], it can be seen as minimizing the strength of a set of edges to be broken minus the number of extra components that are created. This was reduced to  $|E|$  minimum cut problems in [Cunningham, 1985], later it was reduced to  $|V|$  minimum cut problems in [Barahona, 1992]. In this section we discuss important properties of the solutions, and in particular a fast way to update the solution of (6) when a new edge is added to the graph.

**2.1. Properties of optimal partitions.** We start by studying some characteristics of the solutions of the attack problem (6).

**Lemma 1.** *Let  $\Phi = \{S_1, \dots, S_p\}$  be a solution of (6), and let  $\{T_1, \dots, T_q\}$  be a partition of  $S_i$ , for some  $i$ ,  $1 \leq i \leq p$ . Then*

$$w(\delta(T_1, \dots, T_q)) - (q-1) \geq 0.$$

*Proof.* If  $w(\delta(T_1, \dots, T_q)) - (q-1) < 0$  one could improve the solution of (6) by removing  $S_i$  from  $\Phi$  and adding  $\{T_1, \dots, T_q\}$ .  $\square$

**Lemma 2.** *Let  $\Phi = \{S_1, \dots, S_p\}$  be a solution of (6), and let  $\{S_{i_1}, \dots, S_{i_l}\}$  be a sub-family of  $\Phi$ . Then*

$$w(\delta(S_{i_1}, \dots, S_{i_l})) - (l-1) \leq 0.$$

*Proof.* If  $w(\delta(S_{i_1}, \dots, S_{i_l})) - (l-1) > 0$ , one could improve the solution of (6) by removing  $\{S_{i_1}, \dots, S_{i_l}\}$  from  $\Phi$  and adding their union.  $\square$

**Lemma 3.** *There is a unique solution  $\Phi = \{S_1, \dots, S_p\}$  of (6) such that*

$$(7) \quad w(\delta(S_{i_1}, \dots, S_{i_k})) - (k-1) < 0,$$

*for every sub-family  $\{S_{i_1}, \dots, S_{i_k}\}$  of  $\Phi$  with  $k \geq 2$ .*

*Proof.* Let  $\Phi = \{S_1, \dots, S_p\}$  be a solution of (6). If there is a sub-family  $\{S_{i_1}, \dots, S_{i_k}\}$  with  $k \geq 2$  and

$$w(\delta(S_{i_1}, \dots, S_{i_k})) - (k-1) = 0,$$

we replace it by their union. We keep doing this until  $\Phi$  satisfies (7). Let  $\Phi' = \{T_1, \dots, T_q\}$  be another solution of (6).

Assume that there is a set  $T_i$  such that  $T_i \subseteq \cup_{l=1}^k S_{j_l}$ ,  $k \geq 2$ , and  $T_i \cap S_{j_l} \neq \emptyset$  for all  $l$ . Lemma 1 implies  $w(\delta(S_{i_1}, \dots, S_{i_k})) - (k-1) \geq 0$ , a contradiction.

So for each set  $T_i$  there is an index  $j(i)$  such that  $T_i \subseteq S_{j(i)}$ . If for some set  $S_j$  we have  $S_j = \cup_{l=1}^r T_{i_l}$ , with  $r \geq 2$ , Lemmas 1 and 2 imply

$$(8) \quad w(\delta(T_{i_1}, \dots, T_{i_r})) - (r-1) = 0.$$

So  $\Phi'$  violates (7). This proves the uniqueness of  $\Phi$ .  $\square$

Now assume that  $\Phi = \{S_1, \dots, S_p\}$  is a solution of (6) in  $G$ , and a new edge  $e$  is added to produce the graph  $G'$ . A solution of (6) in  $G'$  can be derived from  $\Phi$  as in the next lemma.

**Lemma 4.** *Let  $\Phi = \{S_1, \dots, S_p\}$  be a solution of (6) in  $G$ . Let  $G'$  be the graph obtained by adding one new edge  $e$  to  $G$ . If there is an index  $i$  such that  $e \subseteq S_i$  then  $\Phi$  is a solution of (6) in  $G'$ , otherwise a solution of (6) in  $G'$  is of the form*

$$\Phi' = (\Phi \setminus \{S_i : i \in I\}) \cup \{U = \cup_{i \in I} S_i\},$$

*for some index set  $I \subseteq \{1, \dots, p\}$ , and  $e \in \delta(S_{i_1}, S_{i_2})$ , with  $\{i_1, i_2\} \subseteq I$ . The set  $I$  could be empty, in which case  $\Phi' = \Phi$ . See Figure 1.*

*Proof.* Let  $\{T_1, \dots, T_q\}$  be a solution of (6) in  $G'$ . Assume that there is a set  $S_i$  such that  $S_i \subseteq \cup_{l=1}^k T_{j_l}$ ,  $k \geq 2$ , and  $S_i \cap T_{j_l} \neq \emptyset$  for  $1 \leq l \leq k$ . Lemma 1 implies that

$$w(\delta_{G'}(T_{j_1}, \dots, T_{j_k})) - (k-1) \geq 0,$$

therefore  $\{T_{j_1}, \dots, T_{j_k}\}$  can be replaced by their union. So we can assume that for all  $i$  there is an index  $j(i)$  such that  $S_i \subseteq T_{j(i)}$ .

Now suppose that for some index  $j$ ,  $T_j = \cup_{r=1}^l S_{i_r}$ ,  $l > 1$ . If  $e \notin \delta(S_{i_1}, \dots, S_{i_l})$ , from Lemma 2 we have that

$$w(\delta_{G'}(S_{i_1}, \dots, S_{i_l})) - (l-1) \leq 0,$$

and we could replace  $T_j$  by  $\{S_{i_1}, \dots, S_{i_l}\}$ . If  $e \in \delta_{G'}(S_{i_1}, \dots, S_{i_l})$ , only then we could have

$$w(\delta_{G'}(S_{i_1}, \dots, S_{i_l})) - (l - 1) > 0,$$

and we should keep  $T_j \in \Phi'$ . □

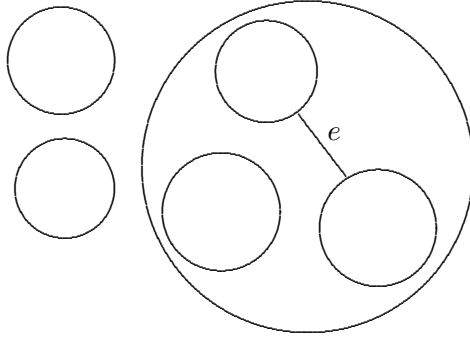


FIGURE 1. The families  $\Phi$  and  $\Phi'$ .

**2.2. Finding an optimal partition.** We give now an algorithm for solving (6). We shall see that this problem is equivalent to optimizing a linear function over a certain *extended polymatroid*. These concepts are discussed in [Grötschel et al., 1993] for instance. This problem can be solved with the greedy algorithm used in [Edmonds, 1970].

Given a graph  $G = (V, E)$  the spanning tree polytope  $T(G)$  is the convex hull of incidence vectors of spanning trees of  $G$ , its dominant is the polyhedron  $P(G) = T(G) + \mathfrak{R}_+^E$  obtained by adding the nonnegative orthant. It follows from [Tutte, 1961] and [Nash-Williams, 1961] that  $P(G)$  is defined by

$$(9) \quad x(\delta(S_1, \dots, S_p)) \geq p - 1, \text{ for every partition of } V,$$

$$(10) \quad x \geq 0.$$

Inequalities (9) are called *partition inequalities*. We reduce problem (6) to the *separation problem* for  $P(G)$ , namely given a vector  $\bar{x} \geq 0$  find a most violated inequality (9) or prove that there is none.

An extended formulation for  $P(G)$  was given in [Jünger and Pulleyblank, 1995] as follows. Associate the variables  $x$  with the edges and the variables  $y$  with the nodes. The system below defines a polyhedron whose projection onto the variables  $x$  is  $P(G)$ . This will be proved at the end of this section. The node  $r$  is an arbitrary element of  $V$ .

$$(11) \quad x(\delta(S)) + y(S) \geq 2, \text{ if } r \notin S, S \subset V,$$

$$(12) \quad x(\delta(S)) + y(S) \geq 0, \text{ if } r \in S, S \subset V,$$

$$(13) \quad y(V) = 0,$$

$$(14) \quad x \geq 0.$$

So given a vector  $\bar{x} \geq 0$  we can try to find a vector  $\bar{y}$  such that  $(\bar{x}, \bar{y})$  satisfies (11)-(14), or prove that  $\bar{y}$  does not exist. Let

$$f(S) = \begin{cases} 2 - \bar{x}(\delta(S)), & \text{if } r \notin S, \\ -\bar{x}(\delta(S)), & \text{if } r \in S, \end{cases}$$

for  $\emptyset \neq S \subseteq V$ . The function  $-f$  is submodular for intersecting pairs, i. e.

$$(15) \quad f(S) + f(T) \leq f(S \cup T) + f(S \cap T),$$

for  $S \neq \emptyset \neq T$ ,  $S, T \subseteq V$ ,  $S \cap T \neq \emptyset$ .

We are going to solve

$$(16) \quad \text{minimize } y(V)$$

subject to

$$(17) \quad y(S) \geq f(S), \text{ for } S \subseteq V.$$

It was shown in [Edmonds, 1970] that the greedy algorithm solves this linear program. This algorithm, which we present below, produces also an optimal solution of the dual problem. We shall see that this gives a most violated partition inequality, if there is any. The dual problem is

$$(18) \quad \text{maximize } \sum z_S f(S)$$

subject to

$$(19) \quad \sum \{z_S | u \in S\} = 1, \text{ for all } u \in V,$$

$$(20) \quad z \geq 0.$$

Given a vector  $y$  satisfying (17), a set  $S$  is called *tight* if  $y(S) = f(S)$ . The function  $y(\cdot) - f(\cdot)$  is nonnegative and submodular for intersecting pairs. So if  $S$  and  $T$  are tight, and  $S \cap T \neq \emptyset$ , then  $S \cap T$  and  $S \cup T$  are also tight.

We start with any vector  $\bar{y}$  satisfying (17), for instance  $\bar{y}(v_i) = 2 \forall i$ , and decrease the value of each  $\bar{y}(v_i)$  until a set becomes tight. We denote by  $\Phi$  the family of tight sets with a positive dual variable. If we try to add  $S$  to  $\Phi$  and there is a set  $T \in \Phi$  with  $S \cap T \neq \emptyset$ , then we replace  $S$  and  $T$  by  $S \cup T$ . This is also tight.

Let  $V = \{v_1, \dots, v_n\}$ , the algorithm is below.

### Algorithm A

- **Step 0.** Start with any feasible vector  $\bar{y}$ , for instance set  $\bar{y}(v_i) \leftarrow 2$  for  $i = 1, \dots, n$ . Also set  $k \leftarrow 1$  and  $\Phi \leftarrow \emptyset$ .
- **Step 1.** If  $v_k$  belongs to a set in  $\Phi$  go to step 3, otherwise set  $\alpha \leftarrow f(\bar{S}) - \bar{y}(\bar{S}) = \max \{f(S) - \bar{y}(S) | v_k \in S\}$ ,  
 $\bar{y}(v_k) \leftarrow \bar{y}(v_k) + \alpha$ ,  
 $\Phi \leftarrow \Phi \cup \{\bar{S}\}$ .
- **Step 2.** While there are two sets  $S$  and  $T$  in  $\Phi$  with  $S \cap T \neq \emptyset$  do  
 $\Phi \leftarrow (\Phi \setminus \{S, T\}) \cup \{T \cup S\}$ .
- **Step 3.** Set  $k \leftarrow k + 1$ , if  $k \leq n$  go to Step 1, otherwise stop.

The vector  $\bar{y}$  is built so it satisfies (17), the family  $\Phi$  defines a partition of  $V$  and  $\bar{y}(S) = f(S)$  for every  $S \in \Phi$ . We set  $\bar{z}_S = 1$ , if  $S \in \Phi$ , and  $\bar{z}_S = 0$  otherwise. We have

$$\bar{y}(V) = \sum \{\bar{y}(S) | S \in \Phi\} = \sum \{f(S) | S \in \Phi\} = \sum \{f(S) \bar{z}_S | S \subseteq V\}.$$

This proves that  $\bar{y}$  and  $\bar{z}$  are optimal solutions.

If the value of the optimum is 0 then  $(\bar{x}, \bar{y})$  satisfies (11), (12), (13). In this case we can pick any partition of  $V$  into  $S_1, \dots, S_p$ , add the inequalities in (11), (12) associated with the sets  $\{S_i\}$ , and  $-y(V) \geq 0$ . We obtain a partition inequality. This shows that  $\bar{x}$  satisfies all the partition inequalities.

Now assume that the value of the optimum is greater than 0. Let  $\bar{z}$  be a 0-1 vector that satisfies the equations (19), the family  $\Gamma = \{S \mid \bar{z}_S = 1\} = \{S_1, \dots, S_p\}$  gives a partition of the set  $V$  and

$$\sum f(S) \bar{z}_S = 2(p-1) - 2\bar{x}(\delta(S_1, \dots, S_p)),$$

so

$$\frac{1}{2} \sum f(S) \bar{z}_S = (p-1) - \bar{x}(\delta(S_1, \dots, S_p)).$$

Since  $\bar{z}$  is an optimum of (18)-(20) it gives a most violated partition inequality.

This procedure shows that (18)-(20) has an optimal integer solution, this is known as the *total dual integrality* of (16)-(17). It also shows that  $\bar{x}$  satisfies (9)-(10) if and only if there is a vector  $\bar{y}$  such that  $(\bar{x}, \bar{y})$  satisfies (11)-(14), so projecting the variables  $y$  in (11)-(14), gives (9)-(10).

This algorithm is easy to modify so it produces the type of solution described in Lemma 3. See below.

**Lemma 5.** *If the set  $\bar{S}$  in Step 1 is chosen so that*

$$f(\bar{S}) - \bar{y}(\bar{S}) = \max \{f(S) - \bar{y}(S) \mid v_k \in S\}, \text{ and } \bar{S} \text{ has maximum cardinality,}$$

*then no uncrossing is needed in Step 2. Moreover the solution satisfies (7), i.e., it is the unique solution described in Lemma 3.*

*Proof.* Since  $\bar{S}$  has maximum cardinality, it should be clear that no uncrossing is needed in Step 2.

Let  $\Phi = \{S_1, \dots, S_p\}$  be the solution obtained, and suppose that

$$\bar{x}(\delta(S_{i_1}, \dots, S_{i_l})) - (l-1) = 0,$$

for a sub-family  $\{S_{i_1}, \dots, S_{i_l}\}$  of  $\Phi$ . Assume that  $r \notin T = \cup_{k=1}^{k=l} S_{i_k}$ , the other case is analogous. We have that  $\bar{y}(S_{i_k}) + \bar{x}(\delta(S_{i_k})) = 2$ , for all  $k$ , then

$$\bar{y}(T) + \bar{x}(\delta(T)) = \sum_{k=1}^{k=l} \bar{y}(S_{i_k}) + \sum_{k=1}^{k=l} \bar{x}(\delta(S_{i_k})) - 2\bar{x}(\delta(S_{i_1}, \dots, S_{i_l})) = 2l - 2(l-1) = 2.$$

Thus the set  $T$  is also tight and it should have been obtained in Step 1.  $\square$

In what follows we shall see that finding the set  $\bar{S}$  in Step 1 reduces to a minimum cut problem. Then it is easy to produce a set of maximum cardinality.

**2.3. Finding tight sets.** It remains to show how to compute the number  $\alpha$  in Step 1. Construct a directed graph  $D = (N, A)$ , where  $N = V \cup \{s, t\}$  and  $A = \{(i, j), (j, i) \mid \{i, j\} \in E\} \cup \{(s, i), (i, t) \mid i \in V\}$ . Define

$$\begin{aligned} \eta(i) &= \bar{y}(i), \text{ for } i \in V, i \neq r, \\ \eta(r) &= \bar{y}(r) + 2, \end{aligned}$$

define capacities

$$\begin{aligned} c(s, i) &= -\eta(i), \quad c(i, t) = 0, \quad \text{if } \eta(i) < 0, i \neq v_k, i \in V, \\ c(i, t) &= \eta(i), \quad c(s, i) = 0, \quad \text{if } \eta(i) \geq 0, i \neq v_k, i \in V, \\ c(s, v_k) &= \infty, \quad c(v_k, t) = \eta(v_k), \quad \text{if } \eta(v_k) \geq 0 \\ c(i, j) &= c(j, i) = \bar{x}(i, j), \quad \text{for } \{i, j\} \in E. \end{aligned}$$

**Lemma 6.** *Suppose that  $\{s\} \cup T$  induces a cut separating  $s$  from  $t$  that has capacity  $\lambda$ , see Figure 2. Then*

$$\bar{y}(T) + \bar{x}(\delta(T)) = \begin{cases} \lambda + \sum\{\eta(v) \mid \eta(v) < 0\} - 2, & \text{if } r \in T \\ \lambda + \sum\{\eta(v) \mid \eta(v) < 0\} & \text{if } r \notin T. \end{cases}$$

*Proof.* Suppose that  $r \in T$ . Then

$$(21) \quad \lambda = \sum\{-\eta(i) \mid i \notin T, \eta(i) < 0\} + \bar{x}(\delta(T)) + \sum\{\eta(i) \mid i \in T, \eta(i) \geq 0\},$$

and

$$\begin{aligned} \lambda + \sum\{\eta(v) \mid \eta(v) < 0\} - 2 &= \\ \sum\{\eta(i) \mid i \in T, \eta(i) < 0\} + \bar{x}(\delta(T)) + \sum\{\eta(i) \mid i \in T, \eta(i) \geq 0\} - 2 &= \\ \bar{y}(T) + \bar{x}(\delta(T)). \end{aligned}$$

The case  $r \notin T$  is analogous. □

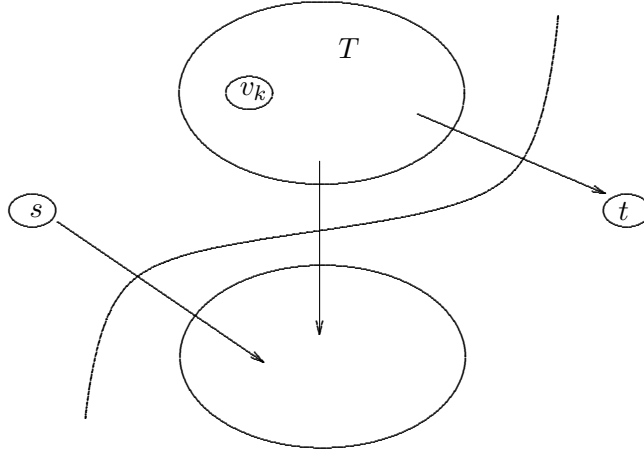


FIGURE 2. The three different types of arcs in a minimum cut.

Therefore if  $\beta$  is the minimum capacity of a cut separating  $s$  from  $t$ , then the value  $\alpha$  is

$$(22) \quad 2 - \beta - \sum\{\eta(v) \mid \eta(v) < 0\}.$$

**2.4. Solving a sequence of related problems.** As we shall see later, we have to solve a sequence of related problems (6) where each time a new edge  $e$  is added to the graph. First notice that the vector  $\bar{y}$  remains feasible in (16)-(17), so one can keep applying the greedy algorithm starting from  $\bar{y}$ . Moreover Lemma 4 tells us that if  $\Phi = \{S_1, \dots, S_p\}$  is the current solution, the new solution will be obtained by taking the union of some sets in  $\Phi$  and leaving the others unchanged.

As described in Section 2.3 the members of  $\Phi$  come from solving a sequence minimum cut problems, each of them associated with each vertex. The complexity of this step dominates the complexity of any other step of the algorithm described in Section 2.2.

We are going to call a *stage* the process of adding a new edge and resolving (6). For each vertex  $u$  let us denote by  $MC_u$  the sequence of minimum cut problems associated with  $u$  over all stages. In the next section we shall see that solving  $MC_u$ , for all  $u$ , can be done in the same asymptotic complexity of  $|V|$  applications of the preflow algorithm of [Goldberg and Tarjan, 1988].

Lemma 4 suggests that in the remaining stages, each set  $S_i$  can be viewed as a single vertex, if so its associated variable is  $y_{S_i} = y(S_i)$ . We do not have to shrink these sets, but for each  $S_i \in \Phi$ , we choose a particular node  $u \in S_i$ , keep working with  $MC_u$ , and discard  $MC_v$  for  $v \in S_i, v \neq u$ . We call  $MC_u$  *active* and  $MC_v$  *inactive* for  $v \in S_i, v \neq u$ . The minimum cut given by  $MC_u$  should be such that  $S_i$  is contained in the source side. This will be discussed in Section 3.2.

Also notice that when adding a new edge  $e$ , any tight set  $S_i$  such that  $e \notin \delta(S_i)$  will remain tight, so there are only two sequences  $MC_u$  that have to be treated at any stage. The following two lemmas give extra information about the structure of the new solution.

**Lemma 7.** *Let  $\Phi = \{S_1, \dots, S_p\}$  be the solution that satisfies (7) and suppose that the edge  $e = \{v, w\}$  is added to the graph, with weight  $\bar{x}(e)$  and  $v \in S_{i_1}, w \in S_{i_2}$ . Suppose that we try to decrease the value  $\bar{y}(v)$  first, let  $T$  be the new maximal tight set encountered. Then either*

- *Case 1:  $T = S_{i_1}$ , or*
- *Case 2:  $S_{i_1} \cup S_{i_2} \subseteq T$ .*

*Proof.* Suppose first that  $e \in \delta(T)$ , then  $T$  was tight before adding  $e$ , since  $S_{i_1}$  was the maximal tight set containing  $v$  then  $S_{i_1} = T$ .

Suppose now that  $v, w \in T$  and that  $\bar{y}(v)$  is being updated as  $\bar{y}(v) \leftarrow \bar{y}(v) - \alpha$ , with  $\alpha \leq \bar{x}(e)$ . Consider the update  $\bar{y}(v) \leftarrow \bar{y}(v) - \alpha/2, \bar{y}(w) \leftarrow \bar{y}(w) - \alpha/2$  and set  $\bar{x}(e) = \alpha/2$ . Then  $S_{i_1}, S_{i_2}$  and  $T$  are tight and  $T' = T \cup S_{i_1} \cup S_{i_2}$  is also tight. But the tightness of  $T'$  does not depend upon the value of  $\bar{x}(e)$ , so  $T' = T$ .  $\square$

**Lemma 8.** *Suppose that Case 1 of Lemma 7 occurs, and that we try to decrease  $\bar{y}(w)$ . Let  $U$  be the new maximal tight set encountered. Then either*

- *Case 1:  $U = S_{i_2}$ , or*
- *Case 2:  $S_{i_1} \cup S_{i_2} \subseteq U$ .*

*Proof.* Suppose first that  $e \in \delta(U)$ , then  $U$  was tight before adding  $e$ , since  $S_{i_2}$  was the maximal tight set containing  $w$  then  $S_{i_2} = U$ .

Suppose now that  $v, w \in U$  and that  $\bar{y}(w)$  is being updated as  $\bar{y}(w) \leftarrow \bar{y}(w) - \beta$ , with  $\beta \leq \bar{x}(e)$ . Notice that  $\bar{y}(v)$  has just decreased by  $\bar{x}(e)$ . Let  $\lambda = \bar{x}(e) + \beta$ . Start with the vector  $\bar{y}$  before adding  $e$  and update as  $\bar{y}(v) \leftarrow \bar{y}(v) - \lambda/2, \bar{y}(w) \leftarrow \bar{y}(w) - \lambda/2$  and set  $\bar{x}(e) = \lambda/2$ . Then  $S_{i_1}, S_{i_2}$  and  $U$  are tight and  $U' = U \cup S_{i_1} \cup S_{i_2}$  is also tight. But the tightness of  $U'$  does not depend upon the value of  $\bar{x}(e)$ , so  $U' = U$ .  $\square$

In a similar way one can prove the Lemma below. This will be required in the next Section.

**Lemma 9.** *Suppose that  $\bar{y}(v)$  has been updated as  $\bar{y}(v) \leftarrow \bar{y}(v) - \alpha$  and  $\bar{y}(w)$  has been updated as  $\bar{y}(w) \leftarrow \bar{y}(w) - \beta$ . Let  $\lambda = (\alpha + \beta)/2$ . Then we can set  $\bar{x}(e) = \lambda$  and starting from the vector  $\bar{y}$  from before adding  $e$ , update as  $\bar{y}(v) \leftarrow \bar{y}(v) - \lambda$ ,  $\bar{y}(w) \leftarrow \bar{y}(w) - \lambda$ . Then the solution of (6) is the same.*

### 3. SOLVING THE SEQUENCE OF MINIMUM CUT PROBLEMS

We shall use the minimum cut algorithm of [Goldberg and Tarjan, 1988] extended in a similar way as in [Gallo et al., 1989]. We first describe the algorithm.

**3.1. The preflow algorithm.** A network is a directed graph  $D = (V, A)$  having a distinguished *source vertex*  $s$ , a distinguished *sink vertex*  $t$ , and a non-negative *capacity*  $c(u, v)$  for each arc  $(u, v)$ . The number of vertices is denoted by  $n$  and the number of arcs by  $m$ . For  $X \subset V$ ,  $s \in X$ ,  $t \notin X$ , we define

$$(X, \bar{X}) = \{(u, v) \in A \mid u \in X, v \notin X\},$$

this is called a *cut*. We extend the capacity function to all vertex pairs by defining  $c(u, v) = 0$  if  $(u, v) \notin A$ . A *flow*  $f$  is a real-valued function on vertex pairs satisfying the following three constraints:

$$(23) \quad f(u, v) \leq c(u, v) \quad \text{for } (u, v) \in V \times V,$$

$$(24) \quad f(u, v) = -f(v, u) \quad \text{for } (u, v) \in V \times V,$$

$$(25) \quad \sum_{u \in V} f(u, v) = 0 \quad \text{for } v \in V \setminus \{s, t\}.$$

A *preflow*  $f$  is a real-valued function on vertex pairs satisfying the capacity constraint (23), the antisymmetry constraint (24), and the following relaxation of the conservation constraint (25):

$$(26) \quad \sum_{u \in V} f(u, v) \geq 0 \quad \text{for } v \in V \setminus \{s, t\}.$$

For a given preflow, the *excess*  $e(v)$  of a vertex  $v$  is  $\sum_{u \in V} f(u, v)$  if  $v \neq s$ , and infinity if  $v = s$ . The *value* of the preflow is  $e(t)$ . A vertex  $v \notin \{s, t\}$  is *active* if  $e(v) > 0$ . A preflow is a flow if and only if  $e(v) = 0$  for all  $v \notin \{s, t\}$ . A vertex pair  $(u, v)$  is a *residual arc* if  $f(u, v) < c(u, v)$ ; the difference  $c(u, v) - f(u, v)$  is the *residual capacity* of the arc. A pair  $(u, v)$  that is not a residual arc is *saturated*. A path of residual arcs is a *residual path*.

A *valid labeling*  $d$  for a preflow  $f$  is a function from the vertices to the nonnegative integers and infinity, such that  $d(t) = 0$ ,  $d(s) = n$ , and  $d(u) \leq d(v) + 1$  for every residual arc  $(u, v)$ . The *residual distance*  $d_f(u, v)$  from a vertex  $u$  to a vertex  $v$  is the minimum number of arcs on a residual path from  $u$  to  $v$ , or infinity if there is no such path. It can be shown that if  $d$  is a valid labeling,  $d(v) \leq \min\{d_f(v, t), d_f(v, s) + n\}$  for any vertex  $v$ .

The preflow algorithm maintains a preflow  $f$ , initially equal to the arcs capacities on arcs leaving  $s$  and zero on arcs not incident to  $s$ . It improves  $f$  by pushing flow excess toward the sink along arcs estimated (by using  $d$ ) to be on the shortest residual paths. As a distance estimate, the algorithm uses a valid labelling  $d$ , initially defined by  $d(s) = n$ ,  $d(v) = 0$  for  $v \neq s$ .

The algorithm uses an *incidence list*  $I(u)$  for each vertex  $u$ . The elements of such a list, called *edges*, are unordered pairs  $\{u, v\}$  such that  $(u, v) \in A$  or  $(v, u) \in A$ . Of the edges on  $I(u)$ , one, initially the first, is designated the *current edge* of  $u$ .

The algorithm consists of repeating the following steps until there are no active vertices:

*Push/Relabel.* Select any active vertex  $u$ . Let  $\{u, v\}$  be the current edge of  $u$ . Apply the appropriate one of the following three cases:

- *Push.* If  $d(u) > d(v)$  and  $f(u, v) < c(u, v)$ , send  $\delta = \min\{e(u), c(u, v) - f(u, v)\}$  units of flow from  $u$  to  $v$ . This is done by adding  $\delta$  to  $f(u, v)$  and  $e(v)$ , and by subtracting  $\delta$  from  $f(v, u)$  and  $e(u)$ . The push is *saturating* if  $\delta = c(u, v) - f(u, v)$  and *nonsaturating* otherwise.
- *Get next edge.* If  $d(u) \leq d(v)$  or  $f(u, v) = c(u, v)$ , and  $\{u, v\}$  is not the last edge on  $I(u)$ , replace  $\{u, v\}$  as the current edge of  $u$  by the next edge on  $I(u)$ .
- *Relabel.* If  $d(u) \leq d(v)$  or  $f(u, v) = c(u, v)$ , and  $\{u, v\}$  is the last edge on  $I(u)$ , replace  $d(u)$  by  $\min\{d(v) \mid \{u, v\} \in I(u) \text{ and } f(u, v) < c(u, v)\} + 1$  and make the first edge on  $I(u)$  the current edge of  $u$ .

When the algorithm terminates,  $f$  is a maximum flow. A minimum cut can be computed as follows. For each vertex  $u$ , replace  $d(u)$  by  $\min\{d_f(u, s) + n, d_f(u, t)\}$  for each  $u \in V$ . The values  $d_f(u, s)$  and  $d_f(u, t)$  for all  $u$  can be computed in  $O(m)$  time. The cut  $(X, \bar{X})$  defined by  $X = \{v \mid d(v) \geq n\}$  is a minimum cut whose source side  $X$  is of maximum cardinality. This is the property required in Lemma 5.

The efficiency of this algorithm depends on the order in which active vertices are selected for push/relabel steps. The following bounds have been given in [Goldberg and Tarjan, 1988].

**Lemma 10.** [Goldberg and Tarjan, 1988]. *Any active vertex  $u$  has  $d_f(u, s) < \infty$ , which implies  $d(u) \leq 2n - 1$ . The value of  $d(u)$  never decreases during the running of the algorithm. The total number of relabel steps is thus  $O(n^2)$ ; together they and all the get next edge steps take  $O(nm)$  time.*

**Lemma 11.** [Goldberg and Tarjan, 1988]. *The number of saturating push steps through any particular residual arc  $(u, v)$  is at most one per value of  $d(v)$ . The total number of saturating push steps is thus  $O(nm)$ .*

So the running time of this algorithm is  $O(nm)$  plus  $O(1)$  time per nonsaturating push steps. In the *first-in first-out* (FIFO) version of the algorithm a queue  $Q$  is used to select vertices for push/relabel steps. Initially every active vertex is appended to  $Q$ . Then the following step is repeated until  $Q$  is empty:

*Discharge.* Remove the vertex  $u$  on the front of  $Q$ . Apply push/relabel steps to  $u$  until  $u$  is no longer active or  $u$  is relabeled. If  $u$  is relabeled, add  $v$  to the rear of  $Q$ . If a push from  $u$  to another vertex  $v$  makes  $v$  active, add  $v$  to the rear of  $Q$ .

**Theorem 12.** [Goldberg and Tarjan, 1988]. *For the FIFO version of the preflow algorithm, the number of nonsaturating push steps is  $O(n^3)$ .*

**3.2. Changing capacities.** Let  $\Phi = \{S_1, \dots, S_p\}$  be the current solution of (6). As described in Section 2.4 a new stage consists of adding a new edge  $\{v, w\}$  and resolving (6). This translates into increasing the capacities of arcs  $(v, w)$  and  $(w, v)$ , for each sequence  $MC_u$ ; notice that each  $MC_u$  has associated a different network.

Assume that after renumbering the sets  $\{S_i\}$ ,  $v \in S_1$  and  $w \in S_2$ . The sets  $S_i$  for  $i > 2$  remain tight, so we have to deal only with  $S_1$  and  $S_2$ . Let  $MC_{u_1}$  be the sequence associated with  $S_1$ . Let  $(X, \bar{X})$  be the current minimum cut, where  $X \setminus \{s\} = S_1$ . Suppose

that the capacity of  $(v, w)$  should increase by  $\alpha$ . For this arc to remain saturated we change  $c(v, w)$  and  $f(v, w)$  as

$$(27) \quad c(v, w) \leftarrow c(v, w) + \alpha,$$

$$(28) \quad f(v, w) \leftarrow f(v, w) + \alpha.$$

In order to have  $e(v) \geq 0$ , we increase  $c(s, v)$  and  $f(s, v)$  as below,

$$(29) \quad c(s, v) \leftarrow c(s, v) + \alpha,$$

$$(30) \quad f(s, v) \leftarrow f(s, v) + \alpha.$$

We have to see that increasing the value of  $c(s, v)$  as in (29) is a valid operation. It follows from Lemma 4 that the new minimum cut should be of the form  $(Y, \bar{Y})$  with  $X \subseteq Y$ , so increasing the capacity of any arc with both endnodes in  $X$  does not change the capacity of  $(Y, \bar{Y})$ . At this point we have a preflow and a valid labeling, so the preflow algorithm can be applied. We call this an *activation*.

Now consider  $S_2$ , and let  $MC_{u_2}$  be the associated sequence. The value  $\bar{y}(v)$  might have decreased. Using the notation of Section 2.3, we have three cases to consider:

- Case 1. If  $\eta(v)$  was nonnegative before the change and remained nonnegative, then  $c(v, t)$  decreases and  $f(v, t)$  is set to  $\min\{f(v, t), c(v, t)\}$ .
- Case 2. If  $\eta(v)$  was negative then  $c(s, v)$  and  $f(s, v)$  increase by the same amount.
- Case 3. If  $\eta(v)$  was nonnegative before the change and became negative, then  $c(v, t)$  and  $f(v, t)$  are set to zero, and  $c(s, v)$  and  $f(s, v)$  increase by some amount  $\gamma$ .

Then the capacity of  $(w, v)$  should increase, so the values  $c(w, v)$ ,  $f(w, v)$ ,  $c(s, w)$  and  $f(s, w)$  are increased as in (27)-(30). Then the preflow algorithm is applied.

Finally we have to deal with a set  $S_i$ ,  $i > 2$ . Let  $MC_{u_i}$  be the associated sequence. Suppose that  $\bar{y}(v)$  has decreased by  $\alpha$  and  $\bar{y}(w)$  has decreased by  $\beta$ . Following Lemma 9 we define  $\lambda = (\alpha + \beta)/2$  and decrease  $\eta(v)$  and  $\eta(w)$  by  $\lambda$  and update  $f$  and  $c$  as in Cases 1, 2 and 3 above. This creates an excess  $e(v) = e(w) = \lambda$ . Then the capacities of  $(v, w)$  and  $(w, v)$  are increased by  $\lambda$ . It might happen that because of the labels  $d(v)$  and  $d(w)$  one of these arcs should be saturated. If so,  $f(v, w)$  or  $f(w, v)$  is increased by  $\lambda$ , then the excess values  $e(v)$  and  $e(w)$  remain nonnegative. We have a preflow and a valid labeling but there is no need to apply the algorithm at this point.

**3.3. Complexity analysis.** First we have to analyze a particular sequence  $MC_u$ . At the beginning every active vertex is appended to  $Q$  and the discharge steps are applied. After each arc activation some new active vertices are added to  $Q$ , and again the algorithm is applied. Let  $l_u$  be the number of activations for  $MC_u$ . The following analysis is similar to the one in [Gallo et al., 1989].

**Theorem 13.** *For a particular node  $u$  and for the entire sequence  $MC_u$ , the FIFO version of the preflow algorithm performs a number of nonsaturating push steps that is  $O(n^3 + l_u n^2)$ .*

*Proof.* Let us define a *pass* over  $Q$ . For each activation, the first pass consists of the discharge steps applied to the vertices initially in  $Q$ . Each pass after the first in an activation consists of the discharge steps applied to the vertices added to  $Q$  during the previous pass. There is at most one nonsaturating push step per vertex  $v$  per pass, since such step reduces  $e(v)$  to zero. It remains to show that the number of passes is  $O(n^2 + l_u n)$ .

Let us define  $\mu = \max\{d(v) \mid v \in Q\}$  if  $Q \neq \emptyset$ , and  $\mu = 0$  if  $Q = \emptyset$ . Consider the effect on  $\mu$  of a pass over  $Q$ . If  $v \in Q$  at the beginning of a pass and  $d(v) = \mu$ , then  $v \notin Q$  at the end of a pass unless a relabel step occurs during the pass. Thus if  $\mu$  is the same as at the end as at the beginning of the pass, some vertex label must have increased by at least one. If  $\mu$  increases over the pass, some vertex label must increase by at least the amount of the increase of  $\mu$ . After each activation  $\mu$  can increase by at most  $n$ . Thus (i) the total number of passes in which  $\mu$  can increase or stay the same is at most  $O(n^2 + l_u)$ ; (ii) the total number of passes in which  $\mu$  can decrease is at most the total increase in  $\mu$  between passes and during passes in which it increases, which is  $O(n^2 + l_u n)$ . Therefore the total number of passes is at most  $O(n^2 + l_u n)$ .  $\square$

Now we analyze the entire collection  $\{MC_u\}$ . First notice that  $\sum_u l_u \leq m$ . Each time one has to compute the numbers  $d_f(u, s)$  and  $d_f(u, t)$ , this takes  $O(m)$  time, so for all sequences this takes  $O(m^2)$  time. Also from the theorem above it follows that the number of nonsaturating push steps for all sequences  $\{MC_u\}$  is  $O(n^4 + mn^2)$ .

So the complexity of solving this entire set of minimum cut problems is  $O(n^4)$ .

#### 4. THE ALGORITHM

Now we can describe the algorithm to solve (3)-(5). As in Section 2, instead of using inequalities (4), we are going to use the extended formulation proposed in [Jünger and Pulleyblank, 1995] as follows. Associate variables  $y$  with the nodes and variables  $x$  with the edges of the graph, choose an arbitrary node  $r$ , and consider the polyhedron  $Q$  defined by

$$\begin{aligned} x(\delta(S)) + y(S) &\geq \begin{cases} 2k & \text{if } r \notin S, \\ 0 & \text{if } r \in S, \end{cases} \\ y(V) &= 0, \\ x &\geq 0. \end{aligned}$$

As already seen in Section 2, the projection of  $Q$  onto the variables  $x$  gives the polyhedron defined by

$$\begin{aligned} x(\delta(S_1, \dots, S_p)) &\geq k(p-1), \text{ for all partitions } \{S_1, \dots, S_p\} \text{ of } V, \\ x(e) &\geq 0. \end{aligned}$$

So we are going to solve the pair of dual linear programs below.

$$(31) \quad \min \sum d_e x_e$$

$$(32) \quad x(\delta(S)) + y(S) \geq \begin{cases} 2k & \text{if } r \notin S, \\ 0 & \text{if } r \in S, \end{cases}$$

$$(33) \quad y(V) = 0,$$

$$(34) \quad -x \geq -u,$$

$$(35) \quad x \geq 0.$$

$$\begin{aligned}
 (36) \quad & \max \sum_{\{S: r \notin S\}} 2kw_S - \sum u_e \beta_e \\
 (37) \quad & \sum_{\{S: e \in \delta(S)\}} w_S \leq d_e + \beta_e \quad \text{for all } e \in E, \\
 (38) \quad & \sum_{\{S: v \in S\}} w_S = \mu \quad \text{for all } v, \\
 (39) \quad & w \geq 0, \quad \beta \geq 0, \quad \mu \text{ unrestricted.}
 \end{aligned}$$

A dual algorithm will be used, i.e., constraints (37), (38) and (39) will always be satisfied and we are going to maximize (36). We start with an informal description.

At the beginning we set to zero all dual variables. We are going to choose a partition  $\{S_1, \dots, S_p\}$  of  $V$  and increase by  $\epsilon$  the value of the variables  $\{w_{S_i}\}$ . This will ensure that constraint (38) is satisfied. We have to ensure that constraints (37) are satisfied for all  $e \in \delta(S_1, \dots, S_p)$ . We say that an edge  $e$  is *tight* if its constraint (37) is satisfied as equation. For a tight edge  $e \in \delta(S_1, \dots, S_p)$  we have to increase the value of  $\beta_e$  by  $2\epsilon$ . Let  $H$  be the subgraph defined by the tight edges. The objective function changes by

$$\epsilon \left( 2k(p-1) - 2u(\delta_H(S_1, \dots, S_p)) \right).$$

So one should find a partition  $\{S_1, \dots, S_p\}$  of  $V$  such that

$$k(p-1) - u(\delta_H(S_1, \dots, S_p)) > 0.$$

Thus we solve

$$(40) \quad \text{minimize } u(\delta_H(S_1, \dots, S_p)) - k(p-1),$$

as described in Section 2. Let  $\Phi = \{S_1, \dots, S_p\}$  be the solution obtained. Let  $(\bar{w}, \bar{\beta}, \bar{\mu})$  be the current dual solution. If the minimum in (40) is negative we use the largest value of  $\epsilon$  so that a new edge becomes tight. This is

$$(41) \quad \bar{\epsilon} = \frac{1}{2} \min \{ \bar{d}_e = d_e - \sum_{\{S: e \in \delta(S)\}} \bar{w}_S \mid e \in \delta_G(S_1, \dots, S_p) \setminus \delta_H(S_1, \dots, S_p) \}.$$

If this minimum is taken over the empty set we say that  $\bar{\epsilon} = \infty$ . In this case the dual problem is unbounded and the primal problem is infeasible.

Now assume that an edge  $e = \{v, q\}$  gives the minimum in (41). If there is more than one edge achieving the minimum in (41) we pick arbitrarily one. We can assume that a perturbation has been done on the edge costs so that the minimum is attained on exactly one edge. Let  $\Phi'$  be the solution of (40) after adding  $e$  to  $H$ . If  $\Phi' = \Phi$  then  $\beta_e$  could increase and  $x_e$  should take the value  $u_e$ , to satisfy complementary slackness; we call this *Case 1*. Otherwise  $\beta_e$  remains equal to zero and  $x_e$  can take a value less than  $u_e$ , this is called *Case 2*. The algorithm stops when the minimum in (40) is zero.

Now we have to describe how to produce a primal solution. At any stage we are going to have a vector  $(\bar{y}, \bar{x})$  satisfying (32), (34), and (35). Equation (33) will be satisfied only at the end.

In order to satisfy complementary slackness for (32), we need that for each set  $S$  with  $\bar{w}_s > 0$  the corresponding inequality holds as equation. Also we can have  $\bar{x}_e > 0$  only if  $e$  is tight, and if  $\bar{\beta}_e > 0$  we should have  $\bar{x}_e = u_e$ .

Initially we set  $\bar{x} = 0$ ,  $\bar{y}_u = 2$  if  $u \neq r$  and  $\bar{y}_r = 0$ . We have to discuss the update of  $(\bar{y}, \bar{x})$  in cases 1 and 2 above.

In Case 1, we set  $\bar{x}_e = u_e$  and update  $\bar{y}$  as  $\bar{y}_v \leftarrow \bar{y}_v - u_e$ ,  $\bar{y}_q \leftarrow \bar{y}_q - u_e$ . Thus for any set  $S$  such that  $e \in \delta(S)$ , if its corresponding inequality (32) was tight, it will remain tight.

**Lemma 14.** *In Case 1 the new vector  $(\bar{y}, \bar{x})$  satisfies (32), (34), and (35).*

*Proof.* Consider an update  $\bar{x}_e = \alpha$ ,  $\bar{y}_v \leftarrow \bar{y}_v - \alpha$ ,  $\bar{y}_q \leftarrow \bar{y}_q - \alpha$ . Let  $\bar{\alpha}$  be the maximum value of  $\alpha$  such that (32) and (34) are satisfied. Notice that the only inequalities (32) that can become tight correspond to sets containing both  $v$  and  $q$ .

If  $\bar{\alpha} < u_e$  there is a new set  $S$  that became tight. Assume that  $v \in S_{i_1} \in \Phi'$  and  $q \in S_{i_2} \in \Phi'$ . Because of the uncrossing arguments of Section 2 we can assume that  $S_{i_1} \subset S$  and  $S_{i_2} \subset S$ . Then we could apply the algorithm of Section 2 starting from  $\bar{y}$  and it would give a solution with  $S_{i_1}$  and  $S_{i_2}$  included in the same member of the partition. A contradiction because the algorithm produces a unique solution independently of the initial feasible point  $\bar{y}$ .  $\square$

In Case 2, we have that

$$\Phi' = (\Phi \setminus \{S_i : i \in I\}) \cup \{U = \cup_{i \in I} S_i\},$$

for some index set  $I \subseteq \{1, \dots, p\}$ , and  $e \in \delta(S_{i_1}, S_{i_2})$ , with  $\{i_1, i_2\} \subseteq I$ . Let

$$(42) \quad \lambda = \begin{cases} \bar{x}(\delta(U)) + \bar{y}(U) - 2k & \text{if } r \notin U, \\ \bar{x}(\delta(U)) + \bar{y}(U) & \text{if } r \in U. \end{cases}$$

We update  $(\bar{y}, \bar{x})$  as  $\bar{y}_v \leftarrow \bar{y}_v - \lambda/2$ ,  $\bar{y}_q \leftarrow \bar{y}_q - \lambda/2$ , and  $\bar{x}_e = \lambda/2$ . Thus the set  $U$  becomes tight.

**Lemma 15.** *In Case 2 the new vector  $(\bar{y}, \bar{x})$  satisfies (32), (34), and (35).*

*Proof.* Consider an update  $\bar{x}_e = \alpha$ ,  $\bar{y}_v \leftarrow \bar{y}_v - \alpha$ ,  $\bar{y}_q \leftarrow \bar{y}_q - \alpha$ . Let  $\bar{\alpha}$  be the maximum value of  $\alpha$  such that (32) and (34) are satisfied.

If  $\bar{\alpha} = u_e < \lambda/2$ . We could apply the algorithm of Section 2 starting from  $\bar{y}$  and it would give a solution where  $U$  is not tight, a contradiction because  $\Phi'$  is the unique solution given by the algorithm.

If  $\bar{\alpha} < u_e$  and  $\bar{\alpha} < \lambda/2$  there is another set that became tight. Again we could apply the algorithm of Section 2 starting from  $\bar{y}$  and it would give a solution where  $U$  is not tight.  $\square$

So at every iteration a new edge becomes tight. In some cases some sets in the family  $\Phi$  are combined into one. When this family consists of only the set  $V$  then we have that  $\bar{y}(V) = 0$  and we have a primal feasible solution that together with the dual solution satisfy complementary slackness. The formal description of the algorithm is below.

### Algorithm B

- **Step 0.** Start with  $\bar{w} = 0$ ,  $\bar{\beta} = 0$ ,  $\bar{\mu} = 0$ ,  $\bar{y}_v = 2k$  if  $v \neq r$ ,  $\bar{y}_r = 0$ ,  $\bar{x} = 0$ ,  $\bar{d}_e = d_e$  for all  $e \in E$ ,  $\Phi$  consisting of all singletons, and  $H = (V, \emptyset)$ .
- **Step 1.** Compute  $\bar{\epsilon}$  as in (41). If  $\bar{\epsilon} = \infty$  stop, the problem is infeasible. Otherwise update  $\bar{w}_{S_i} \leftarrow \bar{w}_{S_i} + \bar{\epsilon}$  for  $S_i \in \Phi$ ,  $\bar{\beta}_e \leftarrow \bar{\beta}_e + 2\bar{\epsilon}$  for all  $e \in \delta_H(S_1, \dots, S_p)$ ,

$$\begin{aligned} \bar{\mu} &\leftarrow \bar{\mu} + \bar{\epsilon}, \\ \bar{d}_e &\leftarrow \bar{d}_e - 2\bar{\epsilon} \text{ for all } e \in \delta_G(S_1, \dots, S_p) \setminus \delta_H(S_1, \dots, S_p). \end{aligned}$$

- **Step 2.** Let  $e$  be an edge giving the minimum in (41), add  $e$  to  $H$ . Solve problem (40) in  $H$  to obtain a partition  $\Phi'$ .
- **Step 3.** If  $\Phi = \Phi'$  update  $(\bar{y}, \bar{x})$  as in Case 1. Otherwise update as in Case 2. If  $\Phi' = \{V\}$  stop, the equation  $\bar{y}(V) = 0$  is satisfied. Otherwise set  $\Phi \leftarrow \Phi'$  and go to Step 1.

It is clear that this algorithm takes at most  $|E|$  iterations. Below we state its complexity.

**Theorem 16.** *The linear program (31)-(35) can be solved in  $O(|V|^4)$  time.*

*Proof.* Notice that to find the minimum in (41) one has to sort the edges only once and then remove the minimum at each iteration. This is because the update of the values  $\bar{d}_e$  does not change their order. So this takes  $O(|E| \log |E|)$  time. All other operations take linear time with the exception of solving (40), that takes  $O(|V|^4)$  over the entire algorithm.  $\square$

It is not difficult to see that when the capacities are infinity the vector  $\bar{x}$  is built in the same way as in Kruskal's algorithm for minimum spanning trees. Also if the data is integer then the primal solution is integer and the dual solution is half-integral as shown below.

**Theorem 17.** *If  $k$  is a nonnegative integer, and the capacities  $u$  are integer, then the vector  $(\bar{y}, \bar{x})$  is integer valued. If  $d$  is integer valued then the dual solution is half-integral.*

*Proof.* Because of the way  $(\bar{y}, \bar{x})$  is being updated, we just have to see that  $\lambda/2$  from (42) is an integer.

Assume that  $r \notin U = \cup_{i \in I} S_i$ , the other case is analogous. We have that  $\bar{y}(S_i) + \bar{x}(\delta(S_i)) = 2$ , for all  $i \in I$ , then

$$\begin{aligned} \lambda &= \bar{y}(U) + \bar{x}(\delta(U)) = \sum_{i \in I} \bar{y}(S_i) + \sum_{i \in I} \bar{x}(\delta(S_i)) - 2\bar{x}(\delta(S_1, \dots, S_l)) = \\ &= 2|I| - 2\bar{x}(\delta(S_1, \dots, S_l)), \end{aligned}$$

so  $\lambda/2$  is an integer.

For the dual problem, if  $d$  is integer valued then  $\bar{\epsilon}$  in (41) is half-integral and  $\bar{d}$  is integral at every iteration. Then the result follows.  $\square$

Consider now the pair of dual linear programs below.

$$(43) \quad \text{minimize } dx$$

subject to

$$(44) \quad x(\delta(S_1, \dots, S_p)) \geq k(p-1), \text{ for all partitions } \Phi = \{S_1, \dots, S_p\} \text{ of } V,$$

$$(45) \quad 0 \leq x(e) \leq u(e).$$

$$\begin{aligned}
(46) \quad & \max \sum \gamma_{\Phi} k(p_{\Phi} - 1) - \sum u_e \beta_e \\
(47) \quad & \sum_{\{\Phi: e \in \delta(S_1, \dots, S_p)\}} \gamma_{\Phi} \leq d_e + \beta_e, \quad \text{for all } e \\
(48) \quad & \gamma \geq 0, \quad \beta \geq 0.
\end{aligned}$$

Here for a partition  $\Phi = \{S_1, \dots, S_p\}$  we define  $p_{\Phi} = p$ . In order to produce a solution for (46)-(48) we start with  $\bar{\gamma} = 0$ , then in Step 1 we add the update  $\bar{\gamma}_{\Phi} \leftarrow \bar{\gamma}_{\Phi} + 2\bar{\epsilon}$ . It should be clear that at any stage the vector  $(\bar{\gamma}, \bar{\beta})$  is feasible and the objective value of  $(\bar{\gamma}, \bar{\beta})$  is the same as the value of  $(\bar{w}, \bar{\beta}, \bar{\mu})$ . Then the theorem below follows.

**Theorem 18.** *The vector  $\bar{x}$  is an optimal solution of (43)-(45). If  $k$  is a nonnegative integer and the capacities  $u$  are integer, then  $\bar{x}$  is integer valued. Also  $(\bar{\gamma}, \bar{\beta})$  is an optimal solution of (46)-(48) and if  $d$  is integer valued then  $(\bar{\gamma}, \bar{\beta})$  is also integer valued.*

**Acknowledgements.** I am grateful to Herve Kerivin for many comments that have improved the presentation.

#### REFERENCES

- [Barahona, 1992] Barahona, F. (1992). Separating from the dominant of the spanning tree polytope. *Op. Research Letters*, 12:201–203.
- [Cheng and Cunningham, 1994] Cheng, E. and Cunningham, W. H. (1994). A faster algorithm for computing the strength of a network. *Inf. Process. Lett.*, 49:209–212.
- [Cunningham, 1985] Cunningham, W. H. (1985). Optimal attack and reinforcement of a network. *J. of ACM*, 32:549–561.
- [Edmonds, 1970] Edmonds, J. (1970). Submodular functions, matroids, and certain polyhedra. In Guy, R. K., Milner, E., and Sauer, N., editors, *Combinatorial Structures and their Applications*, pages 69–87, New York. Gordon and Breach.
- [Gallo et al., 1989] Gallo, G., Grigoriadis, M. D., and Tarjan, R. E. (1989). A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, 18(1):30–55.
- [Goldberg and Tarjan, 1988] Goldberg, A. V. and Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *J. Assoc. Comput. Mach.*, 35(4):921–940.
- [Grötschel et al., 1993] Grötschel, M., Lovász, L., and Schrijver, A. (1993). *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, second edition.
- [Gusfield, 1983] Gusfield, D. (1983). Connectivity and edge-disjoint spanning trees. *Inform. Process. Lett.*, 16(2):87–89.
- [Jünger and Pulleyblank, 1995] Jünger, M. and Pulleyblank, W. R. (1995). New primal and dual matching heuristics. *Algorithmica*, 13:357–380.
- [Nash-Williams, 1961] Nash-Williams, C. S. J. A. (1961). Edge-disjoint spanning trees of finite graphs. *J. London Math. Soc.*, 36:445–450.
- [Roskind and Tarjan, 1985] Roskind, J. and Tarjan, R. E. (1985). A note on finding minimum-cost edge-disjoint spanning trees. *Math. Oper. Res.*, 10(4):701–708.
- [Tutte, 1961] Tutte, W. T. (1961). On the problem of decomposing a graph into  $n$  connected factors. *J. London Math. Soc.*, 36:221–230.

(F. Barahona) IBM T. J. WATSON RESEARCH CENTER, YORKTOWN HEIGHTS, NY 10589, USA

*E-mail address:* `barahon@us.ibm.com`