A LINEAR PROGRAMMING APPROACH TO INCREASING THE WEIGHT OF ALL MINIMUM SPANNING TREES

MOURAD BAÏOU AND FRANCISCO BARAHONA

ABSTRACT. Given a graph where increasing the weight of an edge has a nondecreasing convex piecewise linear cost, we study the problem of finding a minimum cost increase of the weights so that the value of all minimum spanning trees is equal to some target value. Frederickson and Solis-Oba gave an algorithm for the case when the costs are linear, we give a different derivation of their algorithm and we slightly extend it to deal with convex piecewise linear costs. For that we formulate the problem as a combinatorial linear program and show how to produce primal and dual solutions.

Keywords: Minimum weight spanning trees, packing spanning trees, network reinforcement, strength problem.

1. INTRODUCTION

We deal with a graph $G = (V, E)$ where each edge $e \in E$ has an original weight w_e^0 and we can assign to e a new weight $w_e \geq w_e^0$. The *cost* of giving the weight w_e is $c_e(w_e)$. The function $c_e(\cdot)$ is nondecreasing, convex, piecewise linear and $c_e(w_e^0) = 0$, see Figure 1. We study the following problem: Given a value $\lambda \geq 0$ find a minimum cost set of weights so that the weight of a minimum spanning tree is λ.

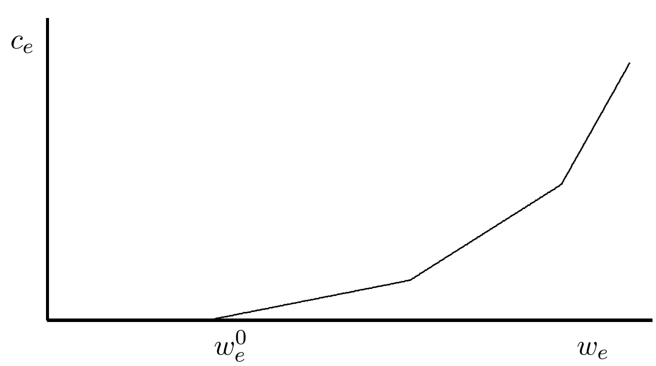

FIGURE 1. Cost of increasing the weight of an edge.

Frederickson and Solis-Oba [6] gave an algorithm for the case when $c_e(\cdot)$ is linear and nondecreasing, here we give a different derivation of their algorithm and we slightly extend it to deal with convex piecewise linear costs. For that we study a linear programming formulation and show how to construct a primal and a dual solution. We also show the relation between this and other combinatorial problems like *network reinforcement* and *packing spanning trees*.

This paper is organized as follows. In Section 2 we give the linear programming formulation. In Section 3 we deal with related combinatorial problems. In Section 4 we describe the network reinforcement problem. In Section 5 we give the algorithm that builds a primal and a dual solution.

The rest of this section is devoted to some definitions and notation. For a family of disjoint node sets $\{S_1, \ldots, S_p\}$ we denote by $\delta(S_1, \ldots, S_p)$ the set of edges with both endpoints in different sets of this family. Sometimes we shall use the notation $\delta_G(S_1, \ldots, S_p)$ to express the fact that this edge set corresponds to edges in G. We are going to write $\delta(v_1, \ldots, v_l)$ instead of $\delta(\{v_1\}, \ldots, \{v_l\})$. For a vector $x \in \mathbb{R}^E$ and a subset $A \subseteq E$, we denote $\sum_{a \in A} x(a)$ by $x(A)$. If $F \subset E$ then $G' = (V, F)$ is called a *spanning subgraph*. If $W \subset V$, and $E(W)$ is the set of edges with both endnodes in W, then $G(W) = (W, E(W))$ is called the *subgraph induced by* W. We denote by n the number of nodes of G, and by m the number of edges of G. We abbreviate "minimum weight spanning tree" by MWST.

2. Increasing the weight of MWSTs: a linear program

For every edge e we have a convex nondecreasing piecewise linear cost function of the weight w_e. This is easy to model using linear programming as follows. Assume that for every edge e there are m_e possible slopes $d_e^1, \ldots, d_e^{m_e}$ of $c_e(\cdot)$. For the value $\bar{w}$ the cost $c_e(\bar{w})$ can be obtained as the optimal value of

$$\min \sum_k d_e^k x_e^k \tag{1}$$

$$\sum_k x_e^k + w_e^0 = \bar{w} \tag{2}$$

$$0 \le x_e^k \le u_e^k, \quad 1 \le k \le m_e. \tag{3}$$

We assume that $d_e^k < d_e^{k+1}$, for $k = 1, \ldots, m_e - 1$. The value u_e^k is the size of the interval for which the slope d_e^k is valid. The solution $\bar{x}$ of this linear program is as follows:

$$\text{there is an index } k_e \ge 1 \text{ such that} \tag{4}$$

$$\bar{x}_e^k = u_e^k, \text{ for } 1 \le k \le k_e - 1, \tag{5}$$

$$u_e^{k_e} > \bar{x}_e^{k_e} = \bar{w} - w_e^0 - \sum_{1 \le k \le k_e - 1} u_e^k \ge 0, \tag{6}$$

$$\bar{x}_e^k = 0, \text{ for } k_e + 1 \le k \le m_e. \tag{7}$$

Thus our problem can be modeled as

$$\min dx \tag{8}$$

$$\sum_{e\in T} w_e \geq \lambda, \text{ for each tree } T \tag{9}$$

$$w_e = w_e^0 + \sum_{k=1}^{m_e} x_e^k, \text{ for each edge } e \tag{10}$$

$$0 \leq x \leq u. \tag{11}$$

This is equivalent to

$$\min dx \tag{12}$$

$$w^0(T) + \sum_{e\in T}\sum_{k=1}^{m_e} x_e^k \geq \lambda, \text{ for each tree } T \tag{13}$$

$$0 \leq x \leq u. \tag{14}$$

This paper is devoted to the study of the linear program (12)-(14) and its connections with other problems from polyhedral combinatorics.

The dual problem is

$$\max \sum_T \big(\lambda - w^0(T)\big) y_T - \sum_{e\in E}\sum_{k=1}^{m_e} \alpha_e^k u_e^k \tag{15}$$

$$\sum_{T\,:\,e\in T} y_T \leq d_e^k + \alpha_e^k, \quad 1 \leq k \leq m_e,\ e \in E \tag{16}$$

$$y, \alpha \geq 0. \tag{17}$$

If $\bar{x}$ is an optimal solution of (12)-(14), it satisfies (5)-(7). So if $(\bar{y}, \bar{\alpha})$ is an optimal solution of (15)-(17), the complementary slackness conditions are as follows: for each edge e let k_e be defined as in (4), then

$$\sum_{T\,:\,e\in T} \bar{y}_T \geq d_e^k, \text{ for } 1 \leq k \leq k_e - 1, \tag{18}$$

$$\text{if } 0 < \bar{x}_e^{k_e} < u_e^{k_e} \text{ then } \sum_{T\,:\,e\in T} \bar{y}_T = d_e^{k_e}, \tag{19}$$

$$\sum_{T\,:\,e\in T} \bar{y}_T \leq d_e^k, \text{ for } k_e \leq k \leq m_e, \tag{20}$$

$$\text{if } \bar{y}_T > 0 \text{ then } \sum_{e\in T}\sum_k \bar{x}_e^k = \lambda - w^0(T). \tag{21}$$

For a weight w_e let $c_e^-(w_e)$ and $c_e^+(w_e)$ be the left-hand and right-hand derivatives of c_e at the value w_e. Notice that $c_e^-(w_e) \leq c_e^+(w_e)$ and the strict inequality holds at the breakpoints. With this notation conditions (18)-(20) can be written as

$$c_e^-(w_e) \leq \sum_{T\,:\,e\in T} \bar{y}_T \leq c_e^+(w_e). \tag{22}$$

3. Related combinatorial problems

In this section we describe Kruskal's algorithm for MWSTs, *packing spanning trees*, and the *attack problem*. All this material will be needed in the following sections.

3.1. **Kruskal's algorithm for MWSTs.**

Assume that we have a graph $G = (V, E)$ with edge weights w_e for $e \in E$, and the weights take values $\omega_1 < \omega_2 < \ldots < \omega_r$. Let

$$F_i = \{e \in E \mid w_e = \omega_i\}.$$

We can describe Kruskal's algorithm for MWSTs as follows. Let $G_1, \cdots, G_p$ be the subgraphs given by the connected components of the spanning subgraph defined by F_1, find a spanning tree in each graph G_i and shrink it to a single node. Repeat the same with F_2 and so on. All MWSTs can be obtained in this way. We illustrate this in Figure 2; the numbers close to the edges are their weights, we also show the nested family of node sets that are being shrunk.

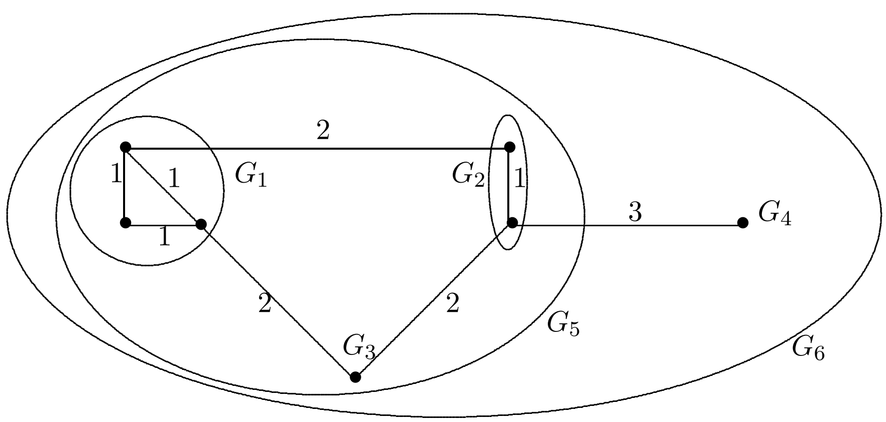

Figure 2. The subgraphs shrunk in Kruskal's algorithm.

We denote by $\{G_i\}$ the family of subgraphs produced by this algorithm.

3.2. **Packing spanning trees.**

Given a graph $G = (V, E)$ with nonnegative edge costs d_e for $e \in E$, we consider the linear program

$$\min dx \tag{23}$$

$$x(T) \geq 1, \text{ for all spanning trees } T \tag{24}$$

$$x \geq 0. \tag{25}$$

Its dual is

$$\max \sum_{T} y_T \tag{26}$$

$$\sum_{T:\, e\in T} y_T \le d_e, \text{ for all } e \in E \tag{27}$$

$$y \ge 0. \tag{28}$$

This last problem can be seen as a *packing of spanning trees* with capacities d. The value of the dual objective function is the *value* of the packing.

Let $\{S_1, \cdots, S_p\}$ be a partition of V, let $\bar{x}$ be a vector defined as

$$\bar{x}_e = \begin{cases} \frac{1}{p-1} & \text{if } e \in \delta(S_1, \cdots, S_p) \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

It follows from the results of [12] and [11] that the extreme points of the polyhedron defined by (24)-(25) are as in (29). Thus solving the linear program (23)-(25) is equivalent to

$$\min \frac{d(\delta(S_1, \cdots, S_p))}{p-1}, \tag{30}$$

where the minimum is taken over all partitions $\{S_1, \cdots, S_p\}$ of V, with $p \ge 2$. This was called the *strength* problem in [5].

It follows from linear programming duality that the value of the minimum in (30) is equal to the value of a maximum packing of spanning trees with capacities d. Also if $\bar{y}$ is a maximum packing and $\{S_1, \cdots, S_p\}$ is a solution of (30), then

$$\sum_{T\,:\, e\in T} \bar{y}_T = d_e$$

for each edge $e \in \delta(S_1, \cdots, S_p)$.

Algorithms for the strength problem have been given in [5], [9], [7] and [4]. The last two references give $O(n^4)$ algorithms. For the dual problem (26)-(28) $O(n^5)$ algorithms have been given in [2] and [8].

The following observation will be used later. Let $\tilde{y}$ be a vector that satisfies (27)-(28). Let $k = \sum_T \tilde{y}_T$, and

$$d'_e = \sum_{T\,:\, e\in T} \tilde{y}_T.$$

Then

$$d'(E) = k(n-1), \tag{31}$$

and

$$d'(\delta(S_1, \cdots, S_p)) \ge k(p-1) \tag{32}$$

for any partition $\{S_1, \cdots, S_p\}$ of V.

3.3. **A simple case.** Here we discuss a simpler version of problem (12)-(14). Later we shall see that the original problem reduces to a sequence of problems of the simpler type.

Assume that every edge has the same original weight w^0 and that x_e is the amount by which the weight can be increased with a per unit cost d_e. Then (12)-(14) can be written as

$$\min dx \tag{33}$$

$$\sum_{e \in T} (w^0 + x_e) \geq \lambda, \text{ for all spanning trees } T \tag{34}$$

$$x \geq 0, \tag{35}$$

or

$$\begin{aligned} &\min dx \\ &\sum_{e \in T} x_e \geq \lambda - (n-1)w^0, \text{ for all spanning trees } T \\ &x \geq 0, \end{aligned}$$

that is equivalent to (23)-(25) when $\lambda > (n-1)w^0$.

3.4. **The attack problem.** Given a set of nonnegative weights u_e for all $e \in E$, and a nonnegative number k consider

$$\min u\big(\delta(S_1, \ldots, S_p)\big) - k(p-1), \tag{36}$$

where the minimization is done over all partitions $\{S_1, \ldots, S_p\}$ of V. Notice that the number p in (36) is not fixed, it is a variable of the problem. This has been called the *attack problem* in [5]. An $O(n^5)$ algorithm was given in [5] and later an $O(n^4)$ was given in [1]. We show here some characteristics of the solutions of the attack problem (36), these appear in [3].

Lemma 1. *Let $\Phi = \{S_1, \ldots, S_p\}$ be a solution of (36), and let $\{T_1, \ldots, T_q\}$ be a partition of S_i, for some i, $1 \leq i \leq p$. Then*

$$u\big(\delta(T_1, \ldots, T_q)\big) - k(q-1) \geq 0.$$

Proof. If $u\big(\delta(T_1, \ldots, T_q)\big) - k(q-1) < 0$ one could improve the solution of (36) by removing S_i from Φ and adding $\{T_1, \ldots, T_q\}$. □

Lemma 2. *Let $\Phi = \{S_1, \ldots, S_p\}$ be a solution of* (36)*, and let $\{S_{i_1}, \ldots, S_{i_l}\}$ be a sub-family of Φ. Then*

$$u\big(\delta(S_{i_1}, \ldots, S_{i_l})\big) - k(l-1) \leq 0.$$

Proof. If $u\big(\delta(S_{i_1}, \ldots, S_{i_l})\big) - k(l-1) > 0$, one could improve the solution of (36) by removing $\{S_{i_1}, \ldots, S_{i_l}\}$ from Φ and adding their union. □

Lemma 3. *If $u(E) = k(n-1)$ and $u(\delta(S_1, \cdots, S_p)) \geq k(p-1)$ for every partition $\{S_1, \cdots, S_p\}$ of V then for $k' \geq k$ a solution of*

$$\min u(\delta(S_1, \cdots, S_p)) - k'(p-1) \tag{37}$$

is the partition of all singletons. The same is true if some edges are deleted before solving (37)*.*

Proof. Since a solution of (36) is the partition of all singletons, it follows from Lemma 2 that

$$u(\delta(v_1, \cdots, v_l)) - k(l-1) \leq 0,$$

for any set of nodes $\{v_1, \cdots, v_l\}$. Therefore

$$u(\delta(v_1, \cdots, v_l)) - k'(l-1) \leq 0. \tag{38}$$

Thus when solving (37), for any partition $\{S_1, \cdots, S_p\}$, if $|S_j| > 1$ it follows from Lemma 1 that one can obtain a partition that is not worse by replacing S_j by all singletons included in S_j. The same is true if some edges are deleted before solving (37). □

Lemma 4. *Let $\Phi = \{S_1, \ldots, S_p\}$ be a solution of (36) 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 Φ is a solution of (36) in G', otherwise a solution of (36) 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 $I \subseteq \{1, \ldots, p\}$, 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 3.

Proof. Let $\{T_1, \ldots, T_q\}$ be a solution of (36) in G'. Assume that there is a set S_i such that $S_i \subseteq \cup_{l=1}^{l=r} T_{j_l}$, $r \geq 2$, and $S_i \cap T_{j_l} \neq \emptyset$ for $1 \leq l \leq r$. Lemma 1 implies that

$$u\big(\delta_G(T_{j_1} \cap S_i, \ldots, T_{j_r} \cap S_i)\big) - k(r-1) \geq 0,$$

and $u\big(\delta_{G'}(T_{j_1}, \ldots, T_{j_r})\big) - k(r-1) \geq 0$. Therefore $\{T_{j_1}, \ldots, T_{j_r}\}$ 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, $T_j = \cup_{q=1}^{q=l} S_{i_q}$, $l > 1$. If $e \notin \delta_{G'}(S_{i_1}, \ldots, S_{i_l})$, from Lemma 2 we have that

$$u\big(\delta_{G'}(S_{i_1}, \ldots, S_{i_l})\big) - k(l-1) \leq 0,$$

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

$$u\big(\delta_{G'}(S_{i_1}, \ldots, S_{i_l})\big) - k(l-1) > 0,$$

and we should keep $T_j \in \Phi'$. □

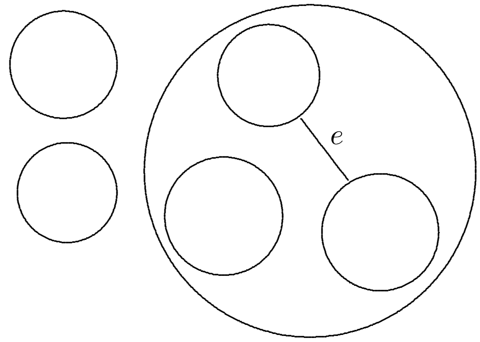

FIGURE 3. The family Φ' is obtained by combining some sets in Φ.

4. Network reinforcement

The *network reinforcement problem* is defined in a graph $G = (V, E)$ with edge costs d, edge capacities u and a nonnegative number k called the *target*. It consists of finding a subgraph of minimum weight that contains k disjoint spanning trees. Multiple copies of each edge can be used; for each edge e the number of copies to be used is bounded by u_e. This problem has been studied in [5], [7] and [3]. The last two references give $O(n^4)$ algorithms. Below we describe the algorithm of [3]. This will be needed in the following section.

For each edge $e \in E$, let x_e be the number of copies of e. If $\{S_1, \dots, S_p\}$ is a partition of V, then x should satisfy

$$x\big(\delta(S_1, \dots, S_p)\big) \geq k(p-1).$$

This set of inequalities is suggested by the results of [12] and [11]. Thus we solve the linear program

$$\text{minimize } dx \tag{39}$$

$$\text{subject to}$$

$$x\big(\delta(S_1, \dots, S_p)\big) \geq k(p-1), \tag{40}$$

$$\text{for all partitions } \Phi = \{S_1, \dots, S_p\} \text{ of } V,$$

$$0 \leq x_e \leq u_e. \tag{41}$$

Instead of using inequalities (40), we use the equivalent extended formulation proposed in [10] as follows. Associate variables y with the nodes and variables x with the edges of the graph, choose an arbitrary node r, and solve the linear program below.

$$\min \sum dx \tag{42}$$

$$x(\delta(S)) + y(S) \geq \begin{cases} 2k & \text{if } r \notin S, \\ 0 & \text{if } r \in S, \end{cases} \quad \text{for all } S \subseteq V, \tag{43}$$

$$y(V) = 0, \tag{44}$$

$$-x \geq -u, \tag{45}$$

$$x \geq 0. \tag{46}$$

Its dual is

$$\max \sum_{\{S:\, r \notin S\}} 2k w_S - \sum u_e \beta_e \tag{47}$$

$$\sum_{\{S:\, e \in \delta(S)\}} w_S \leq d_e + \beta_e \quad \text{for all } e \in E, \tag{48}$$

$$\sum_{\{S:\, v \in S\}} w_S = \mu \quad \text{for all } v, \tag{49}$$

$$w \geq 0, \quad \beta \geq 0, \quad \mu \text{ unrestricted.} \tag{50}$$

A dual algorithm will be used, constraints (48), (49) and (50) will always be satisfied and we are going to maximize (47). For the primal problem, constraints (43), (45), and (46) will always be satisfied, and (44) will be satisfied at the end. Complementary slackness will be kept at all stages. We start with an informal description of the algorithm.

At the beginning we set to zero all dual variables. We are going to choose a partition $\{S_1, \ldots, S_p\}$ of V and increase by ϵ the value of the variables $\{w_{S_i}\}$. This will ensure that constraint (49) is satisfied. We have to ensure that constraints (48) are satisfied for all $e \in \delta(S_1, \ldots, S_p)$. We say that an edge e is *tight* if its constraint (48) is satisfied with equality. For a tight edge $e \in \delta(S_1, \ldots, S_p)$ we have to increase the value of β_e by 2ϵ. Let H be the subgraph defined by the tight edges. The objective function changes by

$$\epsilon\Big(2k(p-1) - 2u\big(\delta_H(S_1, \ldots, S_p)\big)\Big).$$

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

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

Thus we solve

$$\text{minimize } u\big(\delta_H(S_1, \ldots, S_p)\big) - k(p-1), \tag{51}$$

among all partitions $\{S_1, \ldots, S_p\}$ of V. This is problem (36) Let $\Phi = \{S_1, \ldots, S_p\}$ be the solution obtained. Let $(\bar{w}, \bar{\beta}, \bar{\mu})$ be the current dual solution. If the minimum in (51) is negative we use the largest value of ϵ so that a new edge becomes tight. This is

(52)
$$\bar{\epsilon} = \frac{1}{2}\min\Big\{\,\bar{d}_e = d_e - \sum_{\{S:\, e\in\delta(S)\}} \bar{w}_S \mid e \in \delta_G(S_1,\ldots,S_p)\setminus\delta_H(S_1,\ldots,S_p)\Big\}.$$

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. Notice that $\bar{\beta}_e = 0$ if e is not tight, and when an edge becomes tight it remains tight.

Now assume that an edge $e = \{v, q\}$ gives the minimum in (52). If there is more than one edge achieving the minimum in (52) we pick arbitrarily one. Let Φ' be the solution of (51) after adding e to H. If $\Phi' = \Phi$ then β_e could increase and x_e should take the value u_e, to satisfy complementary slackness; we call this *Case 1.* Otherwise according to Lemma 4 we have that

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

for some index set $I \subseteq \{1,\ldots,p\}$, and $e \in \delta(S_{i_1}, S_{i_2})$, with $\{i_1, i_2\} \subseteq I$. If so β_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 (51) 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 (43), (45), and (46). Equation (44) will be satisfied only at the end.

Complementary slackness will be maintained throughout the algorithm. For (43), we need that for each set S with $\bar{w}_S > 0$ the corresponding inequality (43) holds with equality. A set S is called *tight* if the corresponding inequality (43) holds with equality. 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 $\bar{x} = 0$, $\bar{y}_u = 2k$ 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 S was tight it will remain tight.

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 $I \subseteq \{1,\ldots,p\}$, and $e \in \delta(S_{i_1}, S_{i_2})$, with $\{i_1, i_2\} \subseteq I$. Let

(53)
$$\lambda = \begin{cases} \bar{x}(\delta(U)) + \bar{y}(U) - 2k & \text{if } r \notin U, \\ \bar{x}(\delta(U)) + \bar{y}(U) & \text{if } r \in U. \end{cases}$$

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. The new vector satisfies (43), this is shown in [3].

So at every iteration a new edge becomes tight. In some cases some sets in the family Φ 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.

Network Reinforcement

- **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 $e \in E$, Φ consisting of all singletons, and $H = (V, \emptyset)$.
- **Step 1**. Compute $\bar{\epsilon}$ as in (52). 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 $e \in \delta_H(S_1, \ldots, S_p)$,
  $\bar{\mu} \leftarrow \bar{\mu} + \bar{\epsilon}$,
  $\bar{d}_e \leftarrow \bar{d}_e - 2\bar{\epsilon}$ for all $e \in \delta_G(S_1, \ldots, S_p) \setminus \delta_H(S_1, \ldots, S_p)$.
- **Step 2**. Let e be an edge giving the minimum in (52), add e to H. Solve problem (51) in H to obtain a partition Φ'.
- **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.

Since at every iteration a new edge becomes tight, this algorithm takes at most $|E|$ iterations.

5. Producing a primal and a dual vector

5.1. **General Procedure.** We are going to solve (8)-(11) or (12)-(14) as a parametric linear program with parameter λ. First we set $w = w^0$, $x = 0$, $y = 0$, and λ equal to the value of a MWST with weights w^0.

Then we assume that for $\lambda = \lambda^1 \geq 0$ we have an optimal primal solution w^1, and an optimal dual solution (y^1, α^1). We have that if $y^1_T > 0$ then

$$\sum_{e \in T} w^1_e = \lambda^1,$$

and

$$c^-_e(w^1_e) \leq \sum_{T:\, e \in T} y^1_T \leq c^+_e(w^1_e), \tag{54}$$

for each edge e.

Let $\{G_i\}$ be the family of graphs produced by Kruskal's algorithm with weights w^1. In order to increase λ by a small amount we have to increase for some G_i the weight of every MWST. Since all weights in G_i are the same, our problem reduces to (33)-(35) or (23)-(25). So we have to solve (30), where the cost of each edge e is $c^+(w^1_e)$.

For each graph G_i we compute σ_i as the value of the solution of (30). Let $j = \text{argmin}\{\sigma_i\}$, and $\tau = \sigma_j$. Let $\{T^j_1, \cdots, T^j_p\}$ be a partition of V_j that is a solution of (30). Then λ is increased by a small value ϵ and the weights of all edges in $\delta_{G_j}(T^j_1, \cdots, T^j_p)$ is increased by $\epsilon/(p-1)$.

To derive a bound on the value of ϵ, let $\omega_1 < \omega_2 < \ldots < \omega_r$ be the different values of the weights w^1. We set $\omega_{r+1} = \infty$. Let ω_l be the weight of the edges in E_j, $l \leq r$. Let $\rho_1 = \min\{u_e^{k_e} - x_e^{k_e} \,:\, e \in \delta_{G_j}(T_1^j, \cdots, T_p^j)\}$. Let

$$\rho = \min\{\rho_1, \omega_{l+1} - \omega_l\}. \tag{55}$$

Then $0 < \epsilon \leq \rho$.

Now we have to produce a new dual solution that proves the optimality of the new vector w. For that we are going to produce a packing of spanning trees of value τ in each graph G_i and then combine them into a dual vector for the entire graph. First for each graph G_i we are going to compute pseudo costs c' that will be used to find the right packing of spanning trees. For that we solve the network reinforcement problem with target value τ. For each edge e we define

$$c_e^0 = \sum_{T\,:\,e \in T} y_T^1,$$

as its capacity and a cost equal to 1. If $c^+(w_e^1) > c_e^0$ we add a parallel edge with capacity $c^+(w_e^1) - c_e^0$ and cost M, a big number. This problem is feasible because when all capacities are used we obtain a graph that admits a packing of spanning trees of value greater or equal to τ. We need the following lemma.

Lemma 5. *Let c' be the solution of the network reinforcement problem then $c^+(w_e^1) \geq c'_e \geq c_e^0$ for all e.*

Proof. The proof is based on the algorithm of Section 4. It starts with the partition Φ consisting of all singletons. Then the dual variables associated to all sets in Φ take the value $1/2$. Then one edge e with cost 1 becomes tight and its primal variable is set to its upper bound c_e^0. Now we have to see that the algorithm will continue to produce the same partition Φ until all edges with cost 1 become tight.

Let $k = \sum_T y_T^1$ and $k' = \tau$. We have that $k' \geq k$ because k' is the value of a maximum packing of spanning trees in G_j with capacity $c^+(w_e^1)$ for each edge e, and y^1 is a packing that satisfies

$$\sum_{T\,:\,e \in T} y_T^1 \leq c_e^+(w_e^1), \tag{56}$$

for each edge e.

Here $G_i = (V_i, E_i)$ with $V_i = \{v'_1, \cdots, v'_p\}$. We have that

$$c^0(\delta(v'_1, \ldots, v'_p)) = k(p-1)$$

for the trivial partition, see (31); and

$$c^0(\delta(S_1, \ldots, S_q)) \geq k(q-1)$$

for any other partition $\{S_1, \ldots, S_q\}$ of V_i, see (32). Lemma 3 implies that the reinforcement algorithm will use the trivial partition until each edge e with cost 1 becomes tight and its primal variable takes the value c_e^0. Since the algorithm never decreases the value of a variable we have $c'_e \geq c_e^0$.

The definition of the capacities implies $c^+(w_e^1) \geq c'_e$. □

When solving the network reinforcement problem we are minimizing so the solution c' is minimal, thus there is a packing of spanning trees y^i in G_i of value τ and such that

$$\sum_{T:e\in T} y^i_T = c'_e, \text{ for all } e \in E_i. \tag{57}$$

Therefore

$$c^-_e(w^1_e) \leq \sum_{T:e\in T} y^1_T \leq \sum_{T:e\in T} y^i_T \leq c^+_e(w^1_e), \tag{58}$$

for each edge $e \in E_i$.

For the graph G_j we have that

$$\sum_{T:e\in T} y^j_T = c^+_e(w^1_e)$$

for each edge $e \in \delta_{G_j}(T^j_1, \cdots, T^j_p)$. This is because τ is the value of a maximum packing of spanning trees in G_j and all capacities in $\delta_{G_j}(T^j_1, \cdots, T^j_p)$ should be used by this packing.

Then these packings are combined to produce a packing of spanning trees in the original graph as described in the next subsection. This dual solution satisfies the complementary slackness conditions with the new primal solution. This is a proof of optimality.

We can continue increasing λ and the weights of the edges in $\delta_{G_j}(T^j_1, \cdots, T^j_p)$ until one of the following cases arises:

- A breakpoint of the cost function of some edge is found. In this case $\rho = \rho_1$ in (55). This is called a *Type 1* iteration.
- Or $\rho = \omega_{l+1} - \omega_l$ in (55). Here the weights of the edges in $\delta_{G_j}(T^j_1, \cdots, T^j_p)$ reach the value ω_{l+1}. In this case only the family $\{G_i\}$ changes, this is called a *Type 2* iteration.

In either case we restart as in the beginning of this section. If none of these cases is found, i. e. there is no limit for increasing λ, the algorithm stops.

5.2. **Combining Dual Vectors.** Let $\{G_i\}$ be the family of graphs produced by Kruskal's algorithm, we have to describe how to combine the dual vectors produced for each graph G_i. This is done as follows.

Let G_0 be a graph and G_1 an induced subgraph of G_0. Let G_2 be the graph obtained from G_0 by shrinking G_1 to a single node. Assume that we have a packing of spanning trees y^1 of G_1 and a packing y^2 of G_2 both of value τ. We pick $y^1_T > 0$ and $y^2_S > 0$, we set $\mu = \min\{y^1_T, y^2_S\}$ and associate this value to $T \cup S$ that is a tree of the graph G_0. We subtract μ from y^1_T and y^2_S and continue.

This procedure is applied recursively to the family $\{G_i\}$.

5.3. **The algorithm.** Now we can give a formal description of the algorithm:

- **Step 0**. Set $w = w^0$, $x = 0$, $y = 0$, and λ equal to the value of a MWST with weights w^0. Set $k_e = 1$ for each edge e.
- **Step 1**. Let $\omega_1 < \omega_2 < \ldots < \omega_r$ be the different values of the weights w. We set $\omega_{r+1} = \infty$. Let $\{G_i\}$ be the family of graphs produced by Kruskal's algorithm.
  For each graph $G_i = (V_i, E_i)$ compute
  $$\sigma_i = \min c^+(\delta(T_1^i, \cdots, T_p^i), w)/(p-1),$$
  among all partitions $\{T_1^i, \cdots, T_p^i\}$ of V_i.
  Let $j = \text{argmin}\{\sigma_i\}$, and $\sigma_j = c^+(\delta(T_1^i, \cdots, T_p^i), w)/(p-1)$ for a partition $\{T_1^j, \cdots, T_p^j\}$ of V_j. Let ω_l be the weight of the edges in E_j, $l \leq r$.
- **Step 2**. Let $\rho_1 = \min\{u_e^{k_e} - x_e^{k_e} \,:\, e \in \delta_{G_j}(T_1^j, \cdots, T_p^j)\}$,
  $\rho = \min\{\rho_1, \omega_{l+1} - \omega_l\}$. If $\rho = \infty$, stop.
  Otherwise set $x_e^{k_e} \leftarrow x_e^{k_e} + \rho$, $w_e \leftarrow w_e + \rho$, for all $e \in \delta_{G_j}(T_1^j, \cdots, T_p^j)$.
  If $\rho = \rho_1$ go to Step 3, otherwise go to Step 4.
- **Step 3**. Set $k_e \leftarrow k_e + 1$ for all $e \in \delta_{G_j}(T_1^j, \cdots, T_p^j)$ with $x_e^{k_e} = u_e^{k_e}$.
  Produce a new dual solution as described in Subsection 5.1.
- **Step 4**. Set $\lambda \leftarrow \lambda + \rho(p-1)$. Go to Step 1.

5.4. **Complexity Analysis.** Clearly an upper bound for the number of type 1 iterations is $\sum_{e \in E} m_e$. Now we have to derive a bound for the number of type 2 iterations.

Lemma 6. *Between any two iterations of type 1 there are at most* $(n-1)(m-1)$ *iterations of type 2.*

Proof. At any stage of the algorithm there are at most m different values for the edge weights. Let $\omega_1 < \omega_2 \cdots < \omega_r$ be all these values. Let $\rho(\omega_i)$ be the number of edges of weight ω_i in any MWST. At each iteration of type 2 there is an index i such that $\rho(\omega_i)$ decreases and $\rho(\omega_{i+1})$ increases. Thus there are at most $(n-1)(m-1)$ consecutive type 2 iterations. □

At each iteration one has to solve the strength problem (30) for each graph G_i. Let n_i be the number of nodes of G_i. Since this family of node sets is nested, we have that $\sum n_i \leq 2(n-2)$. So the complexity of this sequence of strength problems is $O(n^4)$.

Finding a packing of spanning trees has a complexity $O(n^5)$, with the same arguments as above we have that the complexity of computing the packings for all graphs G_i is $O(n^5)$. We can state the following.

Theorem 1. *The complexity of producing the primal solution is* $O(mn^5 \sum m_e)$, *and the complexity of producing the dual solution is* $O(mn^6 \sum m_e)$.

Acknowledgments

Part of this work was done while the first author was visiting the T. J. Watson Research Center of IBM and while the second author was visiting the Laboratoire d'Econométrie de L'Ecole Polytechnique in Paris. The financial support of both institutions is greatly appreciated.

References

[1] F. Barahona, *Separating from the dominant of the spanning tree polytope*, Op. Research Letters, 12 (1992), pp. 201–203.

[2] F. Barahona, *Packing spanning trees*, Math. Oper. Res., 20 (1995), pp. 104–115.

[3] F. Barahona, *Network reinforcement*, Mathematical Programming, 105 (2006), pp. 181–200. Also in www.optimization-online.org.

[4] E. Cheng and W. H. Cunningham, *A faster algorithm for computing the strength of a network*, Inf. Process. Lett., 49 (1994), pp. 209–212.

[5] W. H. Cunningham, *Optimal attack and reinforcement of a network*, J. of ACM, 32 (1985), pp. 549–561.

[6] G. N. Frederickson and R. Solis-Oba, *Increasing the weight of minimum spanning trees*, J. Algorithms, 33 (1999), pp. 244–266.

[7] H. N. Gabow, *Algorithms for graphic polymatroids and parametric $\overline{s}$-sets*, J. Algorithms, 26 (1998), pp. 48–86.

[8] H. N. Gabow and K. S. Manu, *Packing algorithms for arborescences (and spanning trees) in capacitated graphs*, Math. Programming Ser. B, 82 (1998), pp. 83–109.

[9] D. Gusfield, *Computing the strength of a graph*, SIAM J. Comput., 20 (1991), pp. 639–654.

[10] M. Jünger and W. R. Pulleyblank, *New primal and dual matching heuristics*, Algorithmica, 13 (1995), pp. 357–380.

[11] C. S. J. A. Nash-Williams, *Edge-disjoint spanning trees of finite graphs*, J. London Math. Soc., 36 (1961), pp. 445–450.

[12] W. T. Tutte, *On the problem of decomposing a graph into n connected factors*, J. London Math. Soc., 36 (1961), pp. 221–230.

(M. Baïou) CUST and Laboratoire LIMOS, Université Clermont II, Campus des cézeaux, BP 125, 63173 Aubière cedex, France.

E-mail address, M. Baïou: baiou@custsv.univ-bpclermont.fr

(F. Barahona) IBM T. J. Watson research Center, Yorktown Heights, NY 10589, USA

E-mail address, F. Barahona: barahon@us.ibm.com