New algorithms to compute the strength of a graph

Jérôme Galtier

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

New algorithms to compute the strength of a graph

Jérôme Galtier

N° 6592

July 2008

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--6592--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

New algorithms to compute the strength of a graph

Jérôme Galtier*

Thème COM — Systèmes communicants
Équipe-Projet Mascotte

Rapport de recherche n° 6592 — July 2008 — 14 pages

Abstract: We investigate the problem of computing the strength of a graph. We describe in this article the first polyhedral formulation for the weighted strength in polynomial size of the problem, that is $O(mn)$, where n is the number of vertices and m the number of edges. Moreover, we describe a surprisingly simple FPTAS that gives the strength within $1+\epsilon$ in time $O(m\log^2(n)\log(\frac{m}{n})/\epsilon^2)$ and space $O(m)$, outperforming by a factor of roughly $\min(n\sqrt{m}, n^{5/3})$ the best known exact algorithm of Trubin associated with the Goldberg and Rao maxflow algorithm for that problem, and of roughly $\sigma(G)$ the FPTAS of Plotkin, Shmoys, and Tardos.

Key-words: strength of a graph, matroid, partition, connectivity, community detection, small-world.

* Orange Labs

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Nouveaux algorithmes pour le calcul de la force des graphes

Résumé : Nous nous penchons sur le problème du calcul de la force d'un graphe. Nous décrivons la premième formulation polyhédrale pour le calcul de la force pondérée d'un graphe de taille polynomiale en la taille du problème, à savoir $O(mn)$, où n est le nombre de sommets et m le nombre d'arètes du graphe. Nous décrivons aussi un FPTAS surprenamment simple qui donne la force d'un graphe à un facteur $1+\epsilon$ près en temps $O(m\log^2(n)\log(\frac{m}{n})/\epsilon^2)$ et en espace $O(m)$, dépassant d'un facteur d'environ $\min(n\sqrt{m}, n^{5/3})$ le meilleur algortihme connu de Trubin associé à l'algorithme postérieur de Goldberg and Rao pour le flot maximum et d'environ $\sigma(G)$ le FPTAS de Plotkin, Shmoys, et Tardos.

Mots-clés : force d'un graphe, matroïde, partition, connectivité, détection de communauté, graphe petit-monde

1 Introduction

In this paper we investigate the *strength* of a graph. The notion was introduced by Cunningham to evaluate the resistance of a network under attack [4]. Moreover, it has a strong interest in the field of connectivity and community detection (small world) [8, 13]. We are given a graph $G = (V, E)$, let Π be the set of all partitions over V, and for $P \in \Pi$, δP is the set of edges of G crossing between parts of P (also called the cocycle of P), then the strength is given by

$$\min_{P \in \Pi} \frac{|\delta P|}{|P| - 1}. \tag{1}$$

A weighted notion of strength can also be defined, if $w(e)$ is the positive weight assigned to edge e, and $w(X)$ is the sum of all edges $X \subseteq E$, as:

$$\min_{P \in \Pi} \frac{w(\delta P)}{|P| - 1}. \tag{2}$$

If c is the minimum cut of G, note that the strength will be between $c/2$ and c. An important result on strength of graphs was found by Tutte and Nash-Williams [18, 12]:

Theorem 1 ([18, 12]) *G contains k edge-disjoint spanning trees if and only if the strength of G is larger than or equal to k.*

Most of the previous work that was found to evaluate the strength of a graph depends on the complexity $MF(n, m)$ of finding a maximum $s - t$ flow in a digraph with n vertices and m arcs. The best algorithm we know for that has a complexity of $O(\min(\sqrt{m}, n^{2/3}) m \log(n^2/m + 2))$ [7].

Given a max-flow algorithm, Cunningham gave the first algorithm for the strength, with a complexity of $O(nm\ MF(n, n^2))$ [4]. Later, Gabow and Westermann gave an algorithm to computer the integer value of the strength in $O\left(\min\left(m\sqrt{\frac{m}{n}}(m + n\log n)\log\frac{m}{n}, nm\log\frac{m}{n}\right)\right)$ [6]. Gusfield obtained a $O(n^3 m)$ algorithm[9]. The same year, a FPTAS for that problem is obtained in time $O(m\log^2(n)\sigma(G)/\varepsilon^2)$ by Plotkin, Shmoys and Tardos [14]. The complexity of computing the strength of a graph was also found in $O(n^2\ MF(n, n^2))$ by Barahona [2]. Finally, Trubin, and after Cheng and Cunningham gave an $O(n\ MF(n, m))$ algorithm [17, 3]. For a general perspective on that topic, we point the reader to [15, Chapter 51] and [16].

The article is organized as follows. We recall in Section 2 the basics of the theory of strength of graphs, starting from an alternative definition and showing equivalence to (1). We describe in Section 3 our approximation algorithm and give a proof of correctness. In Section 4 we show an original polyhedral approach of the problem, which is, to our knowledge, the first to be in polynomial size. Note that Sections 3 and 4 are completely independent and therefore Section 3 can be skipped to understand Section 4.

2 Some basic results

We start from an alternative definition for the strength that was used in [1, 15, 16], and prove that it is equivalent respectively to (1) and (2). Let $\mathcal{T}$ be the set of all spanning trees of the graph G.

Definition 1

$$\sigma(G) := \max\left(\sum_{T\in\mathcal{T}} \lambda_T : \forall T \in \mathcal{T}, \lambda_{\mathrm{T}} \geq 0, \forall \mathrm{e} \in \mathrm{E}, \sum_{\mathrm{T}\ni\mathrm{e}} \lambda_{\mathrm{T}} \leq 1\right) \tag{3}$$

and if there are weights on the edges :

$$\sigma_w(G) := \max\left(\sum_{T\in\mathcal{T}} \lambda_T : \forall T \in \mathcal{T}, \lambda_{\mathrm{T}} \geq 0, \forall \mathrm{e} \in \mathrm{E}, \sum_{\mathrm{T}\ni\mathrm{e}} \lambda_{\mathrm{T}} \leq \mathrm{w(e)}\right) \tag{4}$$

By linear duality we can reformulate definition 1 as follows:

Fact 1

$$\sigma(G) = \min\left(\sum_{e\in E} y_e : \forall e \in E, y_e \geq 0, \forall T \in \mathcal{T}, \sum_{\mathrm{e}\in\mathrm{T}} \mathrm{y_e} \geq 1\right), \tag{5}$$

and

$$\sigma_w(G) = \min\left(\sum_{e\in E} w(e)y_e : \forall e \in E, y_e \geq 0, \forall T \in \mathcal{T}, \sum_{\mathrm{e}\in\mathrm{T}} \mathrm{y_e} \geq 1\right). \tag{6}$$

Note that this simple fact can be used in practice to compute the strength of a graph. Computing a minimum spanning tree using the algorithm of Kruskal [10] takes $O(m \log m)$ steps, and since the associated linear program is of size $O(m) \times O(m)$, we can use the claimed result of Fomin [5] to obtain a complexity of $O(m^2 \log^2(m))$ per iteration.

2.1 Properties of the optimal solution

Fact 2 *Let y^* be an optimal solution in (6). For each e with y_e nonzero, there is a tree T_i such that $e \in T_i$ and $\sum_{f\in T} y_f = 1$.*

Proof. Suppose it is not the case. Then for some $\epsilon > 0$, for each T_i with $e \in T_i$ we have $\sum_{f\in T} y^*_f \geq 1 + \epsilon$. Then replacing y^*_e by $\max(0, y_e - \epsilon)$ will improve the optimal solution. □

Fact 3 *Let y^* be an optimal solution in (6). Removing the set $\{e \in E : y^*_e > 0\}$ disconnects G.*

Proof. Suppose on the contrary that the subgraph with the edges e such that $y^*_e = 0$ is connected. Then there is a spanning tree of weight 0, contradicting the definition of y^*. □

Fact 4 *Let y^* be an optimal solution in (6). Let $P = \{C_1, \ldots, C_k\}$ be the connected components obtained by removing the e such that $y^*_e > 0$. Then $\frac{w(\delta P)}{|P|-1} = \sum_{e\in E} w(e)y^*_e$.*

PROOF. Define z_e by $z_e = 1/(|P|-1)$ if $e \in \delta P$, and $z_e = 0$ otherwise. Hence $\forall T \in \mathcal{T} \quad \sum_{e\in T} z_e \geq 1$. By optimality of y^*, $\sum_{e\in E} w(e)y_e^* \leq \sum_{e\in E} w(e)z_e$, and therefore

$$\sum_{e\in E} w(e)y_e^* \leq \frac{w(\delta P)}{|P|-1}. \tag{7}$$

Let $y_{min} = \min_{e:y_e^*>0} y_e^*$. Let T be a tree such that $\sum_{e\in T} y_e^* = 1$. Since T connects all the components of P, $|\delta P \cap T| \geq |P|-1$. So we have

$$1 = \sum_{e\in T} y_e^* \geq (|P|-1)y_{min}.$$

Therefore $y_{min}(|P|-1) \leq 1$. Let us distinguish two cases.

Suppose that $y_{min}(|P|-1) = 1$, then

$$\sum_{e\in E} w(e)y_e^* \geq y_{min}w(\delta P) = \frac{w(\delta P)}{|P|-1},$$

which completes with (7) the fact.

Suppose that, on the reverse, $y_{min}(|P|-1) < 1$, then set for $e \in E$

$$y'_e = \frac{y_e^* - y_{min}\chi(y_e^*>0)}{1-(|P|-1)y_{min}}$$

where $\chi(y_e^*>0)$ equals 1 if $y_e^*>0$, and 0 otherwise. Then, for a tree T on G, we have

$$\sum_{e\in T} y'_e = \frac{\sum_{e\in T} y_e^* - y_{min}|\delta P \cap T|}{1-(|P|-1)y_{min}}. \tag{8}$$

Then construct a set $W = T \cup \{e : y_e^* = 0\}$. Since T is a tree on G, W connects all the vertices of G. We want to construct a tree T' of W with minimum $\sum_{e\in T'} y_e^*$. Since $\delta P = \{e : y_e^* > 0\}$, and $W = T \cup \overline{\delta P}$, we need only $|P|-1$ edges of δP in T' to connect G. So $|T' \cap \delta P| = |P|-1$ and $|(T-T') \cap \delta P| \geq |T \cap \delta P| - |P| + 1$. As a consequence

$$\sum_{e\in T} y_e^* = \sum_{e\in T'} y_e^* + \sum_{e\in T-T'} y_e^* \geq 1 + y_{min}(|T \cap \delta P| - |P| + 1),$$

and replacing in (8), we have $\sum_{e\in T} y'_e \geq 1$. This is true for all tree T, therefore y' satisfies the optimization constraints of (6). Since y^* is an optimal solution,

$$\sum_{e\in E} w(e)y_e^* \leq \sum_{e\in E} w(e)y'_e = \frac{\sum_{e\in E} w(e)y_e^* - y_{min}w(\delta P)}{1-(|P|-1)y_{min}}.$$

So we obtain $\sum_{e\in E} w(e)y_e^* \geq \frac{w(\delta P)}{|P|-1}$, which completes with (7) the fact. □

Fact 5 *We have that*

$$\sigma_w(G) = \min_{Q\in\pi} \frac{w(\delta(Q))}{|Q|-1}.$$

With this fact we can say that the defined $\sigma_w(G)$ corresponds to the weighted strength of (2). Simply setting $w(e) = 1$ shows that $\sigma(G)$ is the strength of (1). Also facts 2, 3 and 4 hold when y^* is the optimal solution of (5).

PROOF. Fact 4 shows that $\sigma_w(G) \geq \min_{Q \in \pi} \frac{w(\delta(Q))}{|Q|-1}$. The reverse is obtained by noticing that, for $Q \in \pi$, we can define z_e by $z_e = 1/(|Q| - 1)$ if $e \in \delta Q$, and $z_e = 0$ otherwise. And we have:

$$\sum_{e \in E} w(e) z_e = \frac{w(\delta Q)}{|Q| - 1} \geq \sigma_w(G).$$

□

2.2 Antitonicity of the strength

We show here that the strength of a graph is always smaller than the strength of its induced subgraphs. We denote $G(S)$ the subgraph induced by the vertices $S \subseteq V$. Note that we have the following important fact:

Fact 6 *Let $P = \{S_1, \ldots, S_p\}$ be a partition of G that achieves the strength of G, that is*

$$\sigma_w(G) = \frac{w(\delta\{S_1, \ldots, S_p\})}{p - 1}$$

then, for all $i \in \{1, \ldots, p\}$, we denote $G(S_i)$ the restriction of G to S_i, and we have:

$$\sigma_w(G(S_i)) \geq \sigma_w(G).$$

PROOF. Indeed suppose on the contrary that there is one partition $S_i^1, \ldots, S_i^k$ of one S_i with

$$\sigma_w(G) > \sigma_w(G(S_i)) = \frac{w(\delta\{S_i^1, \ldots, S_i^k\})}{k - 1}.$$

Then consider the partition $S_1, \ldots, S_{i-1}, S_i^1, \ldots, S_i^k, S_{i+1}, \ldots, S_p$ of G. We have

$$\frac{w(\delta\{S_1, \ldots, S_{i-1}, S_i^1, \ldots, S_i^k, S_{i+1}, \ldots, S_p\})}{k + p - 2}$$

$$= \frac{(p-1)\sigma_w(G) + (k-1)\sigma(G_w(S_i))}{p + k - 2}$$

$$< \sigma_w(G),$$

which is absurd. □

3 A fast approximation

We give in this section a fast algorithm in polynomial time to approximate the strength of a graph. This is inspired from the pre-push flow methods as in [19]. Note that similar method are applied by Plotkin, Shmoys and Tardos [14] but the particular techniques that we use here allow to remove the $\sigma(G)$ factor of this previous work. We allow multiple edges in the following.

Theorem 2 *Given a connected graph G and a positive real $\varepsilon \leq 1/2$, there exists an algorithm of computational time $O(m\log(n)^2\log(\frac{m}{n})/\varepsilon^2)$ and maximum memory space $O(m)$ that returns a set of trees $T_1,\ldots,T_p$ of G, associated to real positive numbers $\lambda_1,\ldots,\lambda_p$ with*

$$\forall e \in E \qquad \sum_{i\in\{1,\ldots,p\}:T_i\ni e} \lambda_i \leq 1 \tag{9}$$

and $\sum_{i\in\{1,\ldots,p\}} \lambda_i \geq \frac{1}{1+\varepsilon}\sigma(G)$.

We introduce the following simple definition coming out of matroid theory.

Definition 2 *A subset F of E is called independent if it contains no cycle.*

According the this definition, the independent subsets of E are the forests and trees are forests of size $n-1$. Then our algorithm, denoted in the following by $\mathcal{A}$ is as follows:

Step 1: Set $\delta := (n(1+\varepsilon))^{-\lceil\frac{3}{\varepsilon}\rceil}(1+\varepsilon)$, $k := \lfloor\frac{m\lceil 1+\log_{1+\varepsilon}(\frac{1}{\delta})\rceil}{n-1}\rfloor$, $T_j := \emptyset$, $t_j := 0$ for $j \in \{1,\ldots,k+1\}$.

Step 2:

For $i := 0$ **to** $\lceil\log_{1+\varepsilon}(\frac{1}{\delta})\rceil$ **do**
  For each edge $e \in E$ **do**
    begin
      Find the minimum $j \in \{1,\ldots,k+1\}$ such that
        $e \notin T_j$ and $T_j \cup \{e\}$ is independent.
      If $j \leq k$ **Then**
        Set $t_j := t_j + 1$
        Set $z(j,t_j) := i$
        Set $T_j := T_j \cup \{e\}$
    end

Step 3: Find the maximum $r \in \{1,\ldots,k\}$ such that $t_r = n-1$.

Step 4: Find the minimum $p \in \{1,\ldots,r\}$ such that $\sum_{l\in\{1,\ldots,n-1\}} \delta(1+\varepsilon)^{z(p,l)} \geq 1$.

Step 5: Let $\forall j \in \{1,\ldots,p\}$ $\qquad \lambda_j := \left(\max_{e\in E} |\{g \in \{1,\ldots,p\} : e \in T_g\}|\right)^{-1}$.

Definition 3 *We say that the independent set $A \subseteq E$ dominates the independent set $B \subseteq E$ if there is no edge in e in B such that $e \notin A$ and $A \cup \{e\}$ is independent.*

Lemma 1 *The algorithm can be run in $O(m)$ space size without increasing the computational time by more than a (small) constant factor.*

PROOF. This can be seen by repeating steps 2 to 3 restricting k over succesive intervals of size of $\lfloor \frac{m}{n} \rfloor$. When an edge cannot be inserted into the current interval, we store the index i in step 2 when it occurs and do not introduce it anymore in step 2 until next interval for k comes. We store that in a list of indexes i when such hits occurs, each i being associated by the edges in question. If steps 3 and 4 succeed to find respectively a proper maximum r and p, then we stop. Otherwise, we resume the computation by considering next interval for k. We then start resuming by the lowest index i for an edge where the computation was stopped, and only introduced edges with such i that have not been used before.

Notice that in each such phase, at least $\frac{m}{n}(n-1)$ edges are required to build the tree (i.e. we perform at least O(m) actual steps of the described algorithm) and the overhead of computation is at most $O(m)$ (constructing the list of passive edges and reintroducing them as active in next phase). □

Lemma 2 *Suppose that A and B are independent sets, and A dominates B. The following assumptions are then true:*

(i) If there is a path in B between vertices u and v, there is also a path in A between u and v.

(ii) If for $e \in E - B$, $B \cup \{e\}$ is not independent then either $e \in A$ or $A \cup \{e\}$ is not independent.

(iii) $|A| \geq |B|$.

PROOF. We show (i) by induction on the length l of the path in B between u and v. If $l = 1$, $\{\{u, v\}\} \in B$, then either $e \in A$ or $A \cup \{\{u, v\}\}$ contains a cycle, and therefore there exists a path in A between u and v. Suppose (i) is true for some l, and there is a path between u and v in B of length $l + 1$. Then there is some w with a path in B between u and w of length l, and $\{w, v\} \in B$. Then u is connected to w by a path in A, and w is connected to v by a path in A, which proves the induction.

Now, $B \cup \{\{u, v\}\}$ is not independent if and only if there is a path in B between u and v, which proves (ii). (iii) comes by induction on $|A|$ by noticing that the number of connected components induced by A is $n - |A|$. □

Lemma 3 *At each time in algorithm $\mathcal{A}$, for $i < j$, T_i dominates T_j, therefore $t_i = |T_i| \geq |T_j| = t_j$.*

PROOF. An edge e can be added to T_j only if $e \in T_i$ or $T_i \cup \{e\}$ is not independent. □

Lemma 4 *Algorithm $\mathcal{A}$ runs in $O(m \log(n)^2 \log(\frac{m}{n})/\varepsilon^2)$ time.*

PROOF.

Step 2: First note that $\log_{1+\varepsilon}(\frac{1}{\delta}) = O(\log(n)/\varepsilon^2)$. So there is $O(m \log(n)/\varepsilon^2)$ iterations in step 2. By lemma 3 a simple dichotomy is enough to find j,

therefore using Kruskal algorithm [15, pp. 98,99,858], required operations for given i and e are up to a constant less than $\log(n)\log(k)$. Also using lemma 1 replacing k by $k_0 = \lfloor \frac{m}{n} \rfloor$ in enough. So an edge insertion is done in $O(\log(n)\log(\frac{m}{n}))$.

Step 3: For a given r, the value t_r is the number of edges in tree T_r. We have $t_r = n-1$ if and only if T_r is a spanning tree. Since a spanning tree is an independent subset of E that dominates all the other independent sets, we can also find T_r by a simple dichotomy. Note also that since $n-1$ edges are necessary to build a spanning tree, and step 2 introduces in all $m\lceil \log_{1+\varepsilon}(\frac{1}{\delta})\rceil$ edges, k is indeed the maximum possible number of spanning trees.

Step 4: During each operation in step 2, t_j is decreasing with j (see lemma 3). Since i is increasing in step 2, for $l \in \{1,\dots,n-1\}$ the quantity $z(j,l)$ is increasing with j. Hence the value

$$v(j) = \sum_{l\in\{1,\dots,n-1\}} \delta(1+\varepsilon)^{z(j,l)},$$

is also increasing with j. Moreover, in the iteration of step 2 where $i = \lceil \log_{1+\varepsilon}(\frac{1}{\delta})\rceil$, all the edges of E are added and therefore at most one tree is completed. This tree T_j receives the weight $\delta(1+\varepsilon)^i$ which is more than 1, and verifies $j \le r$. Therefore p can be found by dichotomy in $O(n\log(k_0)) = O(n\log(\frac{m}{n}))$ computational time.

□

Lemma 5 *For an edge e and an integer $i \in \{0,\dots,\lceil \log_{1+\varepsilon}(\frac{1}{\delta})\rceil\}$ by $\beta(e,i)$ the index j of the tree T_j to which it is added in step 2. Consider*

$$\zeta_j(e) := \delta(1+\varepsilon)^y \text{ where } y = \min\left\{ i \in \{0,\dots,\lceil \log_{1+\varepsilon}(\frac{1}{\delta})\rceil\} : \beta(e,i) \ge j \right\}.$$

Then for $j \in \{1,\dots,p\}$ T_j is a tree of minimum weight on G with weights ζ_j.

PROOF. Note that ζ_j is well defined on each e because $j \le p$.

Consider, from the point of view of T_j, for some j, all the edges that would have been integrated to T_j if $e \notin T_j$ or $T_j \cup \{e\}$ would have been independent. Those are the edges such that at the begining of an iteration i on step 2, there is no $j' < j$ such that $e \notin T_{j'}$ and $T_{j'} \cup \{e\}$ is independent, i.e. those with $\beta(e,i) \ge j$. Therefore ζ_j defines an appropriate weight. Conclude saying that T_j is built with the edges considered in increasing weight, as in the Kruskal algorithm. □

Lemma 6 *We have* $\max_{e\in E} |\{j \in \{1,\dots,p\} : e \in T_j\}| \le \log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right)$.

PROOF. Step 2 can introduce an edge e only $1 + \log_{1+\varepsilon}(\frac{1}{\delta})$ times. □

PROOF. **[Theorem 2].** For each $j \in \{1, \dots, p-1\}$ we have

$$\sum_{e \in E} \zeta_{j+1}(e) = \sum_{e \in E} \zeta_j(e) + \varepsilon \sum_{e \in T_j} \zeta_j(e),$$

so we can conclude by lemma 5 that for $s \in \{1, \dots, p-1\}$,

$$\sum_{e \in E} \zeta_{s+1}(e) - \zeta_1(e) = \varepsilon \sum_{j \in \{1,\dots,s\}} \alpha(\zeta_j),$$

where $\alpha(w) = \min\{w(T) : T \in \mathcal{T}\}$. Note that, by fact 1,

$$\sigma(G) = \min\left\{ \frac{\sum_{e \in E} \zeta(e)}{\alpha(\zeta)} : \zeta \in \mathbb{R}_+^E \right\}.$$

Therefore $\sigma(G) \leq \frac{\sum_{e \in E} \zeta_{s+1}(e) - \zeta_1(e)}{\alpha(\zeta_{s+1} - \zeta_1)}$ and we have $(\alpha(\zeta_{s+1}) - \delta n)\sigma(G) \leq \alpha(\zeta_{s+1} - \zeta_1)\sigma(G) \leq \sum_{e \in E} \zeta_{s+1}(e) - \zeta_1(e)$. It gives:

$$\alpha(\zeta_{s+1}) \leq \delta n + \frac{\varepsilon}{\sigma(G)} \sum_{j \in \{1,\dots,s\}} \alpha(\zeta_j). \tag{10}$$

We now show by induction on κ that

$$\delta n + \frac{\varepsilon}{\sigma(G)} \sum_{j \in \{1,\dots,\kappa\}} \alpha(\zeta_j) \leq \delta n e^{\frac{\kappa\varepsilon}{\sigma(G)}}. \tag{11}$$

It is clear for $\kappa = 0$. Note that for $\kappa > 0$, using the induction hypothesis and (10), we have

$$\begin{aligned} \delta n + \frac{\varepsilon}{\sigma(G)} \sum_{j \in \{1,\dots,\kappa\}} \alpha(\zeta_j) &= \delta n + \frac{\varepsilon}{\sigma(G)} \sum_{j \in \{1,\dots,\kappa-1\}} \alpha(\zeta_j) + \frac{\varepsilon}{\sigma(G)} \alpha(\zeta_\kappa) \\ &\leq \left(1 + \frac{\varepsilon}{\sigma(G)}\right) \delta n e^{\frac{(\kappa-1)\varepsilon}{\sigma(G)}} \end{aligned}.$$

And finaly, equation (11) comes from the fact that $1 + x < e^x$ for $x > 0$.

Putting together (10) and (11) on the particular case of p (step 4), we have:

$$\delta n e^{\frac{p\varepsilon}{\sigma(G)}} \geq \alpha(\zeta_p) \geq 1,$$

which gives $p \geq \frac{\sigma(G)}{\varepsilon} \ln(\frac{1}{n\delta})$. Then we can bound the algorithm as follows:

$$\begin{aligned} \sum_{j \in \{1,\dots,p\}} \lambda_j &= \frac{p}{\max_{e \in E} |\{j \in \{1,\dots,p\} : e \in T_j\}|} \\ &\geq \frac{\frac{\sigma(G)}{\varepsilon} \ln(\frac{1}{n\delta})}{\log_{1+\varepsilon}(\frac{1+\varepsilon}{\delta})} = \frac{\sigma(G) \ln(1+\varepsilon)(\lceil \frac{3}{\varepsilon} \rceil - 1) \ln(n(1+\varepsilon))}{\varepsilon \lceil \frac{3}{\varepsilon} \rceil \ln(n(1+\varepsilon))} = \frac{\sigma(G) \ln(1+\varepsilon)(1 - \frac{1}{\lceil \frac{3}{\varepsilon} \rceil})}{\varepsilon} \\ &\qquad \text{(see lemma 6)} \\ &\geq \frac{\sigma(G)(1-\frac{\varepsilon}{3}) \ln(1+\varepsilon)}{\varepsilon} \geq \frac{\sigma(G)}{1+\varepsilon} \\ &\qquad \left(\text{since for } \varepsilon < \tfrac{1}{2} \qquad \ln(1+\varepsilon) \geq \tfrac{\varepsilon}{(1+\varepsilon)(1-\frac{\varepsilon}{3})}\right) \end{aligned}$$

and we conclude the proof by noticing that (9) is guaranteed by step 5. □

4 A polyhedral approach

We consider the oriented symmetric graph of G, that is, let $\vec{E}$ be the set of arcs derived from E, that is an edge of E gives two arcs in $\vec{E}$. We also denote e the non-oriented version of $\vec{e}$. Let r be an arbitrary vertex of V.

Theorem 3 *The value of $\sigma_w(G)$ is given by the solution of the following linear problem in the real variables $y_e,\ e \in E$, $\gamma_v^k,\ v,k \in V$, $\mu_{\vec{e}}^k,\ k \in V, \vec{e} \in \vec{E}$, and φ:*

$$
\begin{array}{ll}
\textit{Minimize } \sum_{e\in E} w(e)y_e & \\
-\gamma_v^k + \gamma_w^k + \mu_{\overrightarrow{vw}}^k \geq 0, & \forall \overrightarrow{vw} \in \vec{E}, \quad \forall k \in V - \{r\} \\
\varphi - \sum_{k\in V-\{r\}} \mu_{\vec{e}}^k + y_e \geq 0 & \forall \vec{e} \in \vec{E} \\
-\sum_{k\in V-\{r\}} \gamma_r^k + \sum_{k\in V-\{r\}} \gamma_k^k + (n-1)\varphi \leq -1 & \\
\mu_{\vec{e}}^k \geq 0 & \forall \vec{e} \in \vec{E}, \quad \forall k \in V - \{r\} \\
\varphi \geq 0. &
\end{array}
\tag{12}
$$

Moreover the minimum value verifies

$$\sum_{e\in T} y_e \geq 1$$

for any spanning tree T of G.

Proof. We can reformulate fact 1, writing

$$\sigma_w(G) = \min(\sum_{e\in E} w(e)y_e : \forall e \in E, y_e \geq 0, \forall T \in \mathcal{T}, \sum_{\mathrm{e\in E}} \chi_{\{\mathrm{e\in T}\}}\mathrm{y_e} \geq 1), \tag{13}$$

where $\chi_{\{e\in T\}}$ is equal to 1 if $e \in T$, and 0 otherwise.

Consider the set of $\mathbb{R}^E$ given by:

$$\mathcal{S} = \{z \in \mathbb{R}^E \ : \ \exists T \in \mathcal{T}\ \forall e \in E\ z_e = \chi_{\{e\in T\}}\}.$$

Now we can say:

$$\sigma_w(G) = \min\left(\sum_{e\in E} w(e)y_e : \forall e \in E, y_e \geq 0, \forall z \in \mathcal{S}, \sum_{\mathrm{e\in E}} \mathrm{z_e y_e} \geq 1\right), \tag{14}$$

or

$$\sigma_w(G) = \min\left(\sum_{e\in E} w(e)y_e : \forall e \in E, y_e \geq 0, \min_{z\in conv(\mathcal{S})} \sum_{e\in E} z_e y_e \geq 1\right). \tag{15}$$

According to [11, pp.534-535], the set $conv(\mathcal{S})$ can be described as follows. For $\vec{e} \in \vec{E}$ and $k \in V$ we introduce the positive variable $f_{\vec{e}}^k$ of flow of multicommodities.

$$\begin{array}{ll}
\sum_{\vec{e}\in\delta^-(r)} f_{\vec{e}}^k - \sum_{\vec{e}\in\delta^+(r)} f_{\vec{e}}^k = -1 & \forall k \neq r \\
\sum_{\vec{e}\in\delta^-(v)} f_{\vec{e}}^k - \sum_{\vec{e}\in\delta^+(v)} f_{\vec{e}}^k = 0 & \forall v \neq r,\ v \neq k, \forall k \neq r \\
\sum_{\vec{e}\in\delta^-(k)} f_{\vec{e}}^k - \sum_{\vec{e}\in\delta^+(k)} f_{\vec{e}}^k = 1 & \forall k \neq r \\
\sum_{\vec{e}\in\vec{E}} z_{\vec{e}} = n-1 & \\
f_{\vec{e}}^k \leq z_{\vec{e}} & \forall k \neq r\ \forall \vec{e} \in \vec{E}, \\
f_{\vec{e}}^k \geq 0 & \forall k,\ \forall \vec{e} \in \vec{E}.
\end{array} \tag{16}$$

We introduce matrix A and vector b to write problem (16) as $A \cdot \begin{pmatrix} f \\ z \end{pmatrix} \leq b$. Let us fix for a while the values of $y_e,\ e \in E$, and give some $\varepsilon > 0$. What are the conditions for which problem

$$\left\{ \begin{array}{rcl} A \cdot \begin{pmatrix} f \\ z \end{pmatrix} & \leq & b \\ \sum z_e y_e & \leq & 1-\varepsilon \end{array} \right.$$

has no solutions in z and f? Farkas lemma answers that there exists some vector $x_0 \geq 0$ and $\psi \geq 0$ with $x_0^t \cdot A + \psi y = 0$ and $x_0^t \cdot b + (1-\varepsilon)\psi < 0$.

Suppose we have found such a (x_0, ψ) with $\psi = 0$. Then $x_0 \geq 0$ verifies $x_0^t \cdot A = 0$ and $x_0^t \cdot b < 0$, which means, by Farkas lemma again, there is no solution in (f, z) for $A \cdot \begin{pmatrix} f \\ z \end{pmatrix} \leq b$. This implies that the tree polytope of G is empty, which is absurd.

So we have necessarily $\psi > 0$. By division of x_0 by ψ, we see that some x_1 verifies $x_1^t \cdot A + y = 0$ and $x_1^t \cdot b + (1-\varepsilon) < 0$. If this is true for all $\varepsilon > 0$, since the polyhedron $\{x \geq 0 : x^t \cdot A + y = 0\}$ is a closed set, then there also exists a $x_2 \geq 0$ with $x_2^t \cdot A + y = 0$ and $x_2^t \cdot b + 1 \leq 0$.

We have then the following equivalent propositions:

(i)
$$\sum_{e\in E} y_e z_e \geq 1 \qquad \forall z \in \mathcal{S},$$

(ii)
$$\sum_{e\in E} y_e z_e \geq 1 \qquad \forall z \in conv(\mathcal{S}),$$

(iii)
$$\sum_{e\in E} y_e z_e \geq 1 \qquad \forall (z, f) \text{ such that } A \cdot \begin{pmatrix} f \\ z \end{pmatrix} \leq b,$$

(iv) For all $\varepsilon > 0$, there are no solution for

$$\begin{cases} A \cdot \begin{pmatrix} f \\ z \end{pmatrix} & \leq & b \\ \sum z_e y_e & \leq & 1 - \varepsilon, \end{cases}$$

(v) For all $\varepsilon > 0$, there exists a $x \geq 0$ such that

$$\begin{cases} x^t \cdot A + y = 0 \\ x^t \cdot b + (1 - \varepsilon) < 0, \end{cases}$$

(vi) There exists a $x \geq 0$ such that

$$\begin{cases} x^t \cdot A + y = 0 \\ x^t \cdot b + 1 \leq 0. \end{cases}$$

Reformulating the variable x in μ, γ, and φ gives the linear program of theorem 3. Moreover the existence of such a x that verifies $x^t \cdot A + y = 0$ and $x^t \cdot b + 1 \geq 0$, gives that, for any tree T of G, we have

$$\sum_{e \in T} y_e \geq 1.$$

□

Acknowledgements. I would like to thank A. Laugier, J.-C. Bermond, J. Fonlupt and C. Caspar for many helpful remarks. This paper was done in the scope of the joint Orange Labs/INRIA/CNRS/I3S contract Corso2. This paper is also an output of discussions done in the European COST 293 project (GRAAL) and the AEOLUS project.

References

[1] M. Baiou and F. Barahona. A linear programming approach to increasing the weight of all minimum spanning trees. Technical Report Cahier no 2005-12, Ecole Polytechnique - Laboratoire d'Econometrie, May 2005.

[2] F. Barahona. Separating from the dominant of the spanning tree polytope. *Op. Research Letters*, 12:201–203, 1992.

[3] E. Cheng and W. H. Cunningham. A faster algorithm for computing the strength of a network. *Information Processing Letters*, 49:209–212, 1994.

[4] W. H. Cunningham. Optimal attack and reinforcement of a network. *J. of ACM*, 32:549–561, 1985.

[5] S. Fomin. Новый приближенный алгоритм для задачи положительного линейного программирования. *Дискретный анализ и исследование операций*, 2(8):52–72, 2001.

[6] H. Gabow and H. Westermann. Forests, frames and games: algorithms for matroid sums and applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC)*, pages 407–421, 1988.

[7] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *Journal of the ACM*, 45:783–797, 1998.

[8] D. Gusfield. Connectivity and edge-disjoint spanning trees. *Inform. Process. Lett.*, 16(2):87–89, 1983.

[9] D. Gusfield. Computing the strength of a graph. *Siam J. Comput*, 20(4):639–654, 1991.

[10] J. B. Kruskal. On the shortest spanning subtree of a graph and the travelling salesman problem. *Proc. Am. Math. Soc.*, 7(1):48–50, Feb. 1956.

[11] T. Magnanti and L. Wolsey. *Handbooks in OR & MS*, volume 7, chapter Optimal trees. M. O. Ball et al., Eds., Elsevier Science B. V., 1995.

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

[13] Y. Ou and C.-Q. Zhang. Method for data clustering and classification by a graph theory model- network partition into high density subgraphs. International Publication Number WO 2006/119482 A2, November 2006.

[14] S. Plotkin, D. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *IEEE Symposium on Foundations of Computer Science*, pages 495–504, 1991.

[15] A. Schrijver. *Combinatorial optimization*. Springer, 2003.

[16] A. Skoda. *Force d'un graphe, multicoupes et fonctions sous-modulaires: aspects structurels et algorithmiques*. PhD thesis, Université Pierre et Marie Curie, 2007.

[17] V. A. Trubin. Прочность графа и упаковка деревьев и ветвлений. *Кибернетика и Системный Анализ*, 3:94–99, May-June 1993.

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

[19] N. Young. Randomized rounding without solving the linear program. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 170–178, 1995.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399