RC 20677(91344)(11/19/96)
Computer Science / Mathematics

IBM Research Report

On the k-cut Problem

Francisco Barahona

IBM Research Division
T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598

ON THE K-CUT PROBLEM

FRANCISCO BARAHONA†

Abstract. Given a graph with nonnegative edge-weights, let $f(k)$ be the value of an optimal solution of the k-cut problem. We study f as a function of k. Let g be the convex envelope of f. We give a polynomial algorithm to compute g. In particular, if f is convex, then it can be computed in polynomial time for all k. We show some experiments in computing g.

Key words. k-cut problem, submodular functions, minimum cut, clustering

AMS subject classifications. 90C27, 05C85

1. Introduction. Let $G = (V, E)$ be a graph with nonnegative weights $w(e)$ for each edge e. Given a partition $S_1, \ldots, S_k$ of V, the set of edges with endnodes in different sets of the partition is called a *k-cut*. This is denoted by $\delta(S_1, \ldots, S_k)$. We use $w(T)$ to denote $\sum_{e \in T} w(e)$. We use n to denote $|V|$. For a given value of k the *k-cut problem* consists of finding a partition $S_1, \ldots, S_k$ that minimizes

$$w\big(\delta(S_1, \ldots, S_k)\big). \tag{1.1}$$

The k-cut problem is a clustering problem. The edge weights represent similarity between objects. Then the problem can be stated as finding k clusters that maximize the similarity between objects in the same cluster.

The k-cut problem is NP-Hard if k is part of the input. If k is fixed, Goldschmidt and Hochbaum [8] showed that this reduces to $O(n^{k^2})$ minimum cut problems.

Other problems dealing with multicuts where k is not fixed have also been studied. Chvátal [5] considered

$$\text{minimize } w\big(\delta(S_1, \ldots, S_p)\big)/p, \tag{1.2}$$

where p is variable. He showed that the minimum is achieved for $p = 2$, i. e. by a minimum cut.

The closely related problem

$$\text{minimize } w\big(\delta(S_1, \ldots, S_p)\big)/(p-1), \tag{1.3}$$

reduces to n parametric min-cut computations, see [4], [6]. Although the two problems look very similar, the solution of (1.3) is much more involved than the one for (1.2).

Consider now

$$\text{minimize } w\big(\delta(S_1, \ldots, S_p)\big) - p, \tag{1.4}$$

where p is variable, and $p \geq 2$. This is called the *multicut problem*. It was shown in [1] that this reduces to $O(n^3)$ minimum cut problems. The methods used for (1.4) are very much related

†IBM T. J. Watson Research Center, P. O. 218, Yorktown Heights, New York 10598.

to the ones for (1.3). One can think of (1.4) as a relaxation of (1.1), we explore that idea in this paper.

Let $f(k)$ be the value of an optimal solution of the k-cut problem. We say that f is convex if

$$f(k+1) - f(k) \leq f(k+2) - f(k+1),$$

for $2 \leq k \leq n-2$. The *convex envelope* g of f, is the largest convex function such that $g(k) \leq f(k)$, for $2 \leq k \leq n$. In this paper we show that g can be computed in polynomial time, using (1.4) as a subroutine.

This paper is organized as follows. In Section 2 we sketch the solution approach for the multicut problem. In Section 3 we show how to compute the convex envelope g. In Section 4 we present some experiments with this approximation.

2. The multicut problem. Problem (1.4) reduces to minimizing a submodular function as shown in [1]. In this section we sketch that reduction.

We need first a few definitions. A function $f : 2^E \to \Re$ is called submodular if

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B) \text{ for all } A, B \subseteq E.$$

The concept of a submodular function in discrete optimization is in many respects analogous to that of a convex function in continuous optimization. The only known polynomial algorithm to find the minimum of a submodular function is based on the ellipsoid method, cf. [9]. However Queyranne [12] gave recently a simple combinatorial algorithm for minimizing symmetric submodular functions. A function f is *symmetric* if $f(A) = f(E \backslash A)$, for all $A \subseteq E$.

Given $A \subseteq V$, we use $\delta(A)$ to denote the set of edges with exactly one endnode in A, and $\gamma(A)$ is the set of edges with both endnodes in A. It is well known that the function $f(S) = w(\delta(S))$, for $S \subseteq V$, is submodular.

Consider

$$m(S) = w(\delta(S)) - 2 + h(S),$$

where

$$h(S) = \min \left\{ w(\delta(T_1, ..., T_q)) - (q-1) \right\},$$

and $T_1, ..., T_q$ is a partition of S, $1 \leq q \leq |S|$. Then problem (1.4) reduces to minimizing $m(S)$ for $\emptyset \neq S \subset V$. Now we need the following two Lemmas.

LEMMA 2.1. (*Cunningham [6]*). *Let $F = \gamma(S)$, and r be the rank function of the graphic matroid. One can compute $h(S)$ by adding $1 - |S|$ to the value of*

$$\text{(2.2)} \qquad \begin{cases} \textit{maximize } y(F) \\ \textit{subject to} \\ y(T) \leq r(T), \textit{ if } T \subseteq F, \\ y \leq w. \end{cases}$$

Proof. For $A \subseteq F$, let $\bar{A} = F \backslash A$. We have

$$y(F) = y(A) + y(\bar{A}) \leq w(A) + r(\bar{A}).$$

Given a vector $\bar{y}$ feasible for (2.2), the *greedy algorithm* cf. [7] picks any component $\bar{y}(e)$ and increases its value until it reaches its upper bound $w(e)$, or one of the other inequalities becomes tight. In the latter case we say that a set is *tight*. Let $\hat{y}$ be obtained by applying the greedy algorithm. Then for any $e \in F$ either $\hat{y}(e) = w(e)$ or e is in a tight set. Let $\bar{A}$ be the union of the tight sets. This is also tight, by submodularity. Thus

$$\hat{y}(E) = \hat{y}(A) + \hat{y}(\bar{A}) = w(A) + r(\bar{A}).$$

This shows that the solution of (2.2) gives the minimum of

$$w(A) + r(\bar{A}).$$

By taking $A = \emptyset$, we have that the minimum is at most $|S| - 1$. So it is equivalent to look for the minimum of

$$w(A) + r(\bar{A}) - (|S| - 1) = w(F) - (|S| - 1) + r(\bar{A}) - w(\bar{A})$$

that is nonpositive.

It is easy to see that it is enough to take sets $T = \gamma(B)$ in (2.2). So $\bar{A}$ is the union of sets of type $\gamma(B)$. We are going to obtain the minimum of

$$\sum_{i=1}^{l} \Big[(|T_i| - 1) - w(\gamma(T_i))\Big] + w(F) - (|S| - 1).$$

By taking $T_{l+i} = \{v_i\}$ for $v_i \in S \setminus \bigcup_{1 \le j \le l} T_j$, we have

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

□

LEMMA 2.3. *The function h is submodular.*

Proof. We have to consider (2.2) for $F = \gamma(S)$, $F = \gamma(T)$, $F = \gamma(S \cup T)$ and $F = \gamma(S \cap T)$. Suppose that $\bar{y}$ was obtained after applying the greedy algorithm for $F = \gamma(S \cap T)$. We can extend $\bar{y}$ to a solution for $F = \gamma(S \cup T)$, denote this by $\bar{\bar{y}}$. Now denote by $\bar{\bar{y}}_S$ the vector obtained from $\bar{\bar{y}}$ by setting to zero all components not in $\gamma(S)$. We can extend $\bar{\bar{y}}_S$ to a solution for $F = \gamma(S)$. We can proceed similarly for $F = \gamma(T)$. This shows that

$$h(S \cup T) + h(S \cap T) \le h(S) + h(T).$$

□

The function m is not symmetric, however we can define

$$m'(S) = \frac{1}{2}\, w(\delta(S)) - 1 + h(S),$$

and minimize $m'(S) + m'(\bar{S})$ that is symmetric. Queyranne's algorithm requires $O(n^3)$ evaluations of m'. In our case one evaluation of m' requires $O(n)$ minimum cut problems, see [2]. So the straightforward implementation would require $O(n^4)$ minimum cuts. However one can use the solution of a previous step to start the next one, this improves the bound to $O(n^3)$ minimum cut problems, see [1].

3. The convex envelope. Given k, a lower bound of $f(k)$ is

$$l(\lambda, k) = \min_{p \geq 2} w\big(\delta(S_1, \ldots, S_p)\big) - \lambda(p - k),$$

where $\lambda \geq 0$. The function $l(\cdot, k)$ is concave and piecewise linear. Its maximum

$$g(k) = \max_{\lambda \geq 0} l(\lambda, k)$$

is also a lower bound for $f(k)$. Now we have to see how to compute g.

The function

$$l'(\lambda) = \min_{p \geq 2} w\big(\delta(S_1, \ldots, S_p)\big) - \lambda p,$$

is piecewise linear and concave, for $\lambda \geq 0$. Its slope takes values in $\{-2, \ldots, -n\}$. Now we describe how to obtain l' for $\lambda \geq 0$. Given $0 \leq \lambda_1 < \lambda_2$, we can evaluate l' by solving two multicut problems, let p_1 and p_2 be the sizes of the partitions obtained. If $p_1 = p_2$, there is no breakpoint in $[\lambda_1, \lambda_2]$, and we have obtained l' for this interval. If $p_1 < p_2$, we have to obtain all breakpoints in $[\lambda_1, \lambda_2]$. we compute λ_3 as the solution of

$$t = l'(\lambda_1) - p_1(\lambda - \lambda_1) = l'(\lambda_2) - p_2(\lambda - \lambda_2)$$

and compute $l'(\lambda_3)$.

If $l'(\lambda_3) = t$ then λ_3 is the only breakpoint in $[\lambda_1, \lambda_2]$, and we are done with this interval. If $l'(\lambda_3) < t$, we have obtained a new slope p_3 at λ_3, and $p_1 < p_3 < p_2$. Then we repeat the same procedure for $[\lambda_1, \lambda_3]$ and $[\lambda_3, \lambda_2]$. Each time we find a new slope or a new breakpoint, so $2(n-1)$ is a bound for the number of multicut problems that we have to solve. The first interval should be given by $\lambda_1 = 0$, and a large number like $\lambda_2 = \sum w(e)$.

Assume now that we know all breakpoints $\{\alpha_1, \ldots, \alpha_r\}$ of l', and slopes $\{-p_1, \ldots, -p_{r+1}\}$. Then for any k, it is easy to compute $g(k)$. Since $l(\cdot, k)$ is concave and piecewise linear, its maximum is at a point with zero slope, or at a breakpoint where the slope changes from positive to negative. So we have to find p_i and p_{i+1} with $p_i \leq k \leq p_{i+1}$. Then

$$g(k) = l'(\alpha_i) + \alpha_i k.$$

Suppose now that we use linear interpolation to extend g to noninteger arguments. We obtain a convex piecewise linear function whose breakpoints are $\{p_2, \ldots, p_r\}$, with slopes $\{\alpha_1, \ldots, \alpha_r\}$.

Now let us see how well g approximates f. We have that $g(2) = f(2)$, $g(n) = f(n)$, and $g(k) \leq f(k)$ for any other value $k \in \{2, \ldots, n\}$. Also g is convex, piecewise linear, and if k is a breakpoint of g then $g(k) = f(k)$. So g is the best convex approximation of f, i. e. its convex envelope. Now we can state our main result.

THEOREM 3.1. *The computation of the convex envelope of f reduces to $O(n^4)$ minimum cut problems. Moreover, if f is convex then f can be computed in polynomial time.* ☐

4. Some experiments. We decided to compute g for some practical instances. The main issue here was to see how this function looks, other aspects like efficiency of the implementation or CPU time are beyond the scope of this paper.

We used some Network Design instances from [3]. In this case, the edge weights represent traffic between cities. A first step in the design process is to define the *backbone network*. This is exactly a clustering problem. Once the clusters are chosen, the backbone network is designed

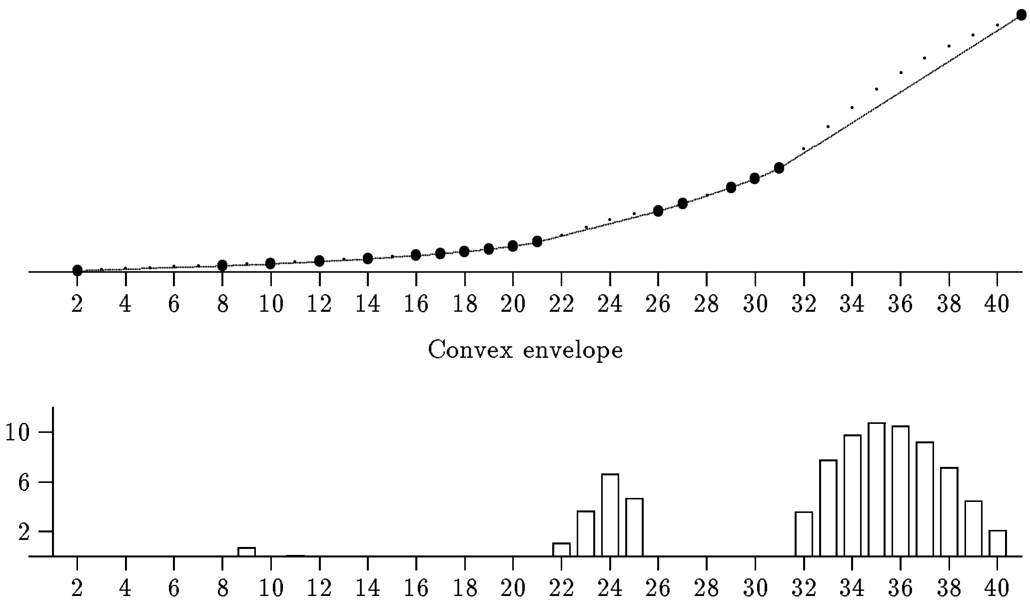

FIG. 1. *A network with 41 nodes.*

to connect them. Finally the *local networks* are designed to connect cities within each cluster. See [11].

In each case we computed the function g and an upper bound for f. For the upper bound we used the three heuristics below.

- Min-cut heuristic. Given a k-cut, delete its edges. Find a minimum cut in each component. Find the minimum value among all these min cuts. Break this component to derive a $(k+1)$-cut.
- Largest edge heuristic. Start with the n-cut consisting of each individual node. Find an edge of maximum weight. Contract this edge to obtain an $(n-1)$-cut. Continue with the shrunken graph.
- Gomory-Hu heuristic. Compute the Gomory-Hu tree associated with the graph. Delete $k-1$ edges of minimum weight. The remaining connected components define a k-cut. It was shown in [10] that this heuristic has a performance guarantee of $2(1-1/k)$.

We first tried a network with 41 cities. We plot in Figure 1 the convex envelope and the upper bound for each value of k. The breakpoints of g are marked by a bullet ($\bullet$). We also plot the gap (in percentage) between g and the upper bound. The largest gap in this case was 10%.

We did a similar experiment with a 47 cities network, see Figure 2. This time the largest gap was 24%. The function f seems to be concave in the interval [2, 43]. This is reflected in the larger gap.

We also tried the 64 cities network of [3], see Figure 3. This time the largest gap was 2.5%. This seems to reflect the "more convex shape" of the function f.

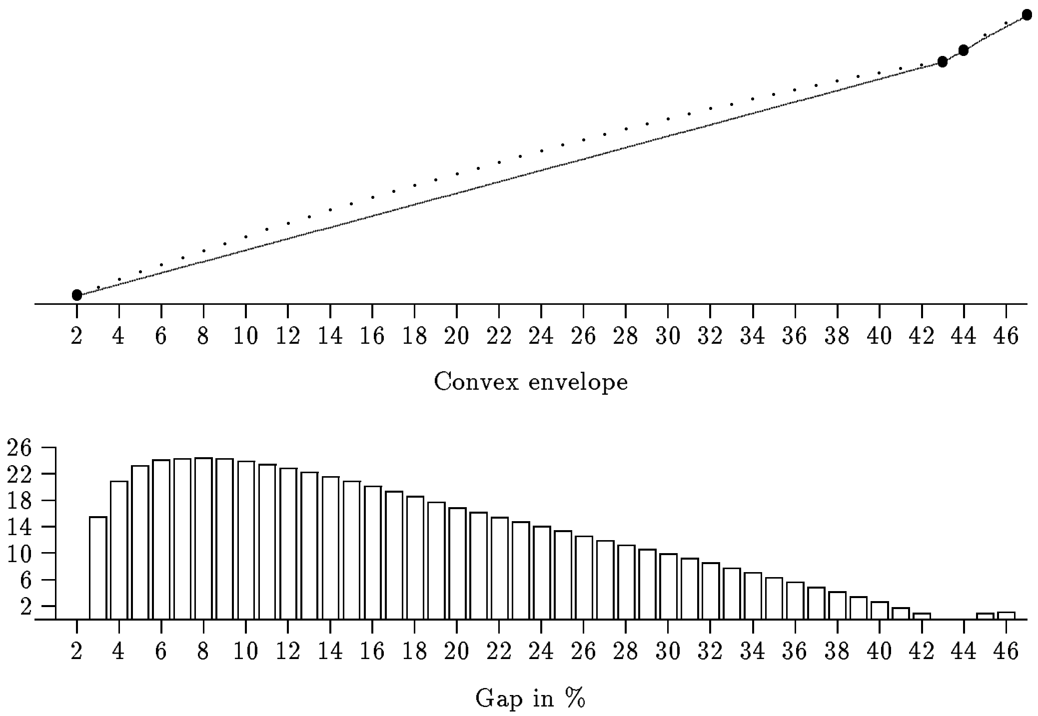

FIG. 2. *A 47 nodes network.*

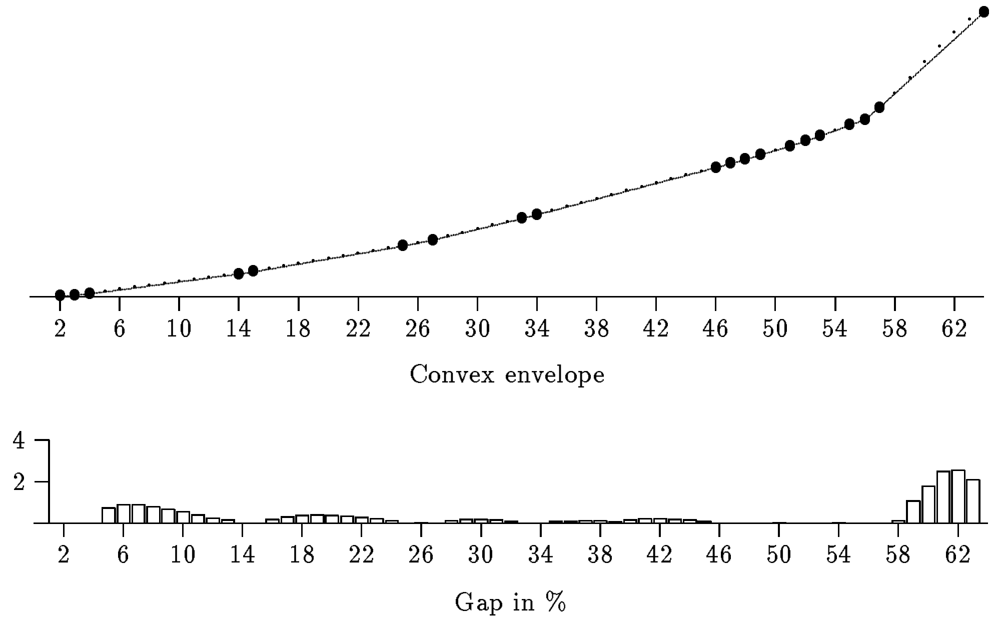

FIG. 3. *A 64 nodes network.*

Finally we tried two graphs randomly generated. In Figure 4 we present the results given by a graph with 50 nodes, 7% density and edge weights taking integer values in [0, 100]. Figure 5 comes from a graph with 50 nodes, 50% density and similar edge weights. These two instances seem to indicate that this approximation works better for sparse graphs. If the graph is a tree then clearly f is convex. It would be very interesting to characterize other classes of graphs for which f is convex, or where the gap between f and g is bounded.

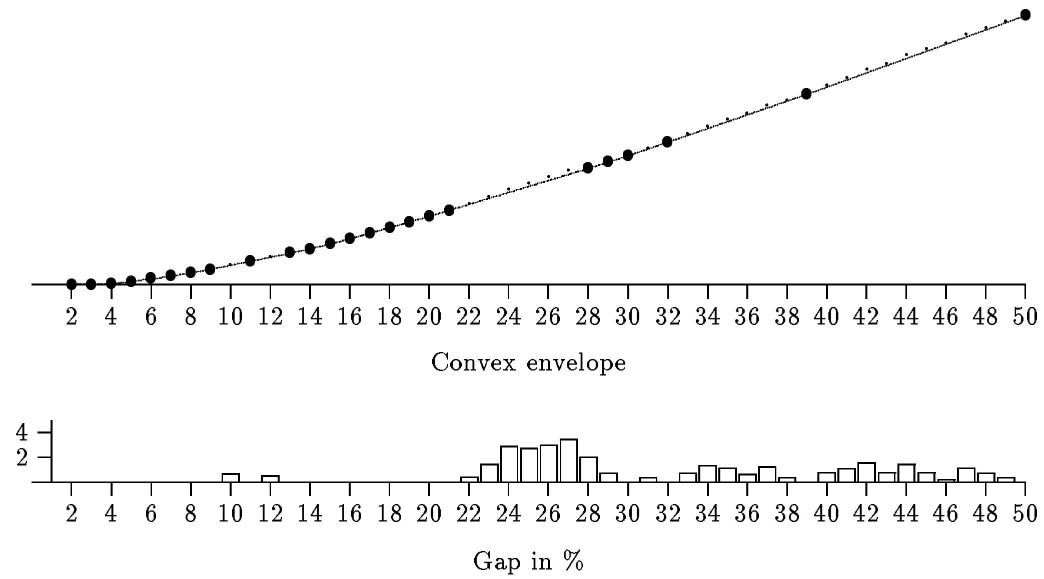

FIG. 4. *A graph with 7% density.*

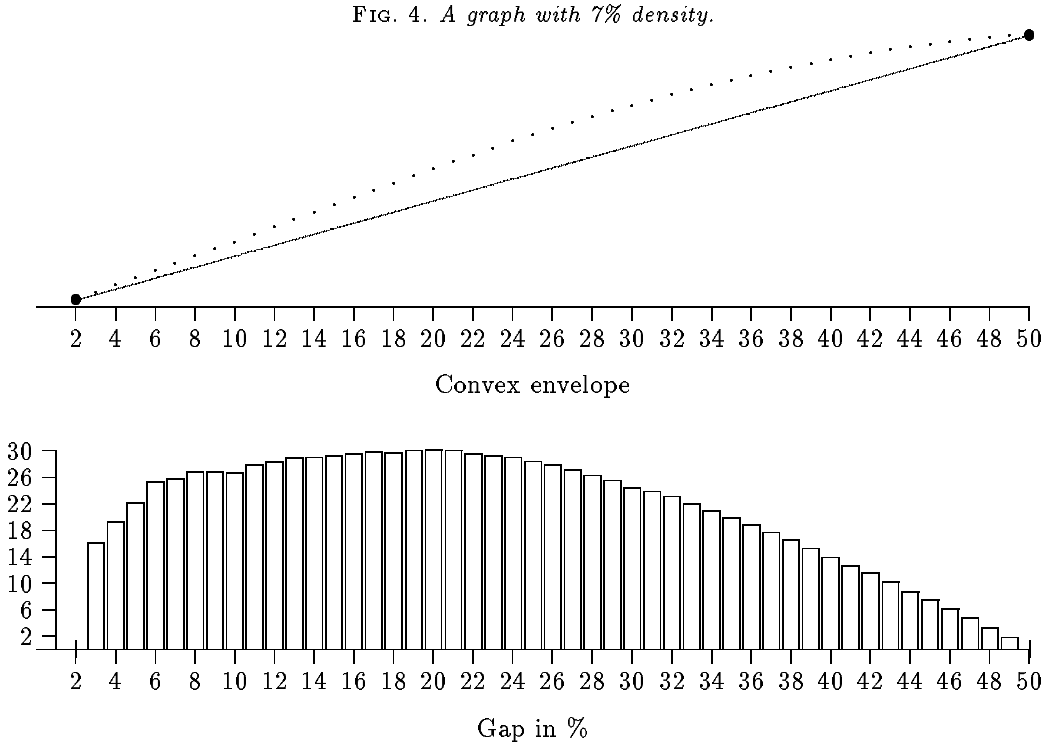

FIG. 5. *A graph with 50% density.*

REFERENCES

[1] M. BAÏOU, F. BARAHONA, AND A. R. MAHJOUB, *Separation of partition inequalities*, Report, 1996.

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

[3] ———, *Network design using cut inequalities*, SIAM Journal on Optimization, 6 (1996), pp. 823–837.

[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] V. Chvátal, *Tough graphs and hamiltonian circuits*, Discrete Mathematics, 5 (1973), pp. 215–228.

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

[7] J. Edmonds, *Submodular functions, matroids, and certain polyhedra*, in Combinatorial Structures and their Applications, R. K. Guy, E. Milner, and N. Sauer, eds., New York, 1970, Gordon and Breach, pp. 69–87.

[8] O. Goldschmidt and D. S. Hochbaum, *A polynomial algorithm for the k-cut problem for fixed k*, Math. Oper. Res., 19 (1994), pp. 24–37.

[9] M. Gröstchel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer-Verlag, Berlin, 1988.

[10] S. Kapoor, *On minimum 3-cuts and approximating k-cuts using cut trees*, in Integer Programming and Combinatorial Optimization, LNCS 1084 Springer-Verlag, 1996, pp. 132–146.

[11] A. Kershenbaum, *Telecomunications network design algorithms*, McGraw-Hill, New York, 1993.

[12] M. Queyranne, *A combinatorial algorithm for minimizing symmetric submodular functions*, in Proc. 6th ACM-SIAM Symp. Discrete Algorithms, 1995, pp. 98–101.