C. De Simone, G. Rinaldi

A CUTTING PLANE ALGORITHM FOR THE MAX-CUT PROBLEM

Caterina De Simone, Giovanni Rinaldi - Istituto di Analisi dei Sistemi ed Informatica del CNR - Viale Manzoni 30, 00185 Roma, Italy.

This work was partially supported by EEC Contract SC1-CT-91-0620.

Abstract

In this paper we describe a cutting plane algorithm to solve max-cut problems on complete graphs. We show that the separation problem over the cut polytope can be reduced to the separation problem over the cut cone and we give a separation algorithm for a class of inequality valid over the cut cone: *the hypermetric inequalities*. Computational results are given.

Key words: Polyhedral Combinatorics, Separation Problem, Maximum Cut Problem, Hypermetric Inequality.

1. Introduction

For every graph G with vertex set V and edge set E, let $K(G)$ denote the set of all vectors x such that

$$x_{ij} = \begin{cases} 1 & \text{if precisely one of } i,j \text{ belongs to } S \\ 0 & \text{otherwise,} \end{cases}$$

for some (possibly empty) subset S of V; the *cut* corresponding to S is the set $\delta(S)$ of edges with exactly one endpoint in S. (In particular, we allow $S = \emptyset$, in which case $\delta(S)$ is a zero vector.) The problem

$$\begin{aligned} &\max\ c^T x \\ &\text{s.t. } x \in K(G), \end{aligned}$$

with $c \in \mathbb{Z}^E$, is called the *max-cut problem* in G. This problem is equivalent to the linear programming problem

$$\begin{aligned} &\max\ c^T x \\ &\text{s.t. } x \in P_C(G), \end{aligned} \tag{1.1}$$

where $P_C(G)$ denotes $\mathrm{conv}(K(G))$, the convex hull of $K(G)$, and is called the *cut polytope* of G.

An integer linear programming formulation of (1.1) is:

$$\begin{aligned} &\max\ c^T x \\ &\text{s.t. } \sum_{ij\in D} x_{ij} - \sum_{ij\in C-D} x_{ij} \le |D| - 1 \text{ for each } (C,D) \in T(G), \\ &\quad 0 \le x_{ij} \le 1 \quad \text{for each } ij \in E', \\ &\quad x_{ij} \text{ integer} \quad \text{for each } ij \in E, \end{aligned} \tag{1.2}$$

where $T(G)$ denotes the set of all ordered pairs (C,D) such that C is the edge set of a chordless cycle in G and D is an odd-cardinality subset of C, and where E' denotes the set of all the edges of G that belong to no triangle.

For every $(C,D) \in T(G)$, the inequality

$$\sum_{ij\in D} x_{ij} - \sum_{ij\in C-D} x_{ij} \le |D| - 1$$

is called a *cycle inequality*. Note that, for each triangle ijk contained in G there are precisely four corresponding cycle inequalities, namely

$$\begin{aligned} x_{ij} + x_{ik} + x_{jk} &\leq 2 \\ x_{ij} - x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} + x_{ik} - x_{jk} &\leq 0 \\ -x_{ij} - x_{ik} + x_{jk} &\leq 0. \end{aligned} \tag{1.3}$$

Each inequality in (1.3) is called a *triangle inequality*.

In [3], Barahona, Jünger, and Reinelt describe a cutting plane algorithm to solve problem (1.2) that uses cycle inequalities as cutting planes.

In this paper we describe an algorithm for the solution of max-cut problems on complete graphs. For these graphs the only chordless cycles are triangles, and so problem (1.2) reads:

$$\begin{aligned} &\max\ c^T x \\ &\text{s.t. } x \in T_n, \\ &\qquad x_{ij} \text{ integer} \quad \text{for each } ij \in E, \end{aligned} \tag{1.4}$$

where T_n denotes the set of all points in $\mathbb{R}^{\binom{n}{2}}$ satisfying all triangle inequalities (1.3). T_n is called *triangle polytope*. Hence, when applied to a complete graph, the algorithm proposed in [3] uses only triangle inequalities as cutting planes.

To exploit a stronger relaxation of (1.1), our algorithm uses triangle inequalities as cutting planes as well as other inequalities valid over the cut polytope of a complete graph, the so called *hypermetric inequalities*.

For every integer vector $b = [b_1, \cdots, b_n]^T$ such that $b_1 + \cdots + b_n = 1$, the hypermetric inequality specified by the vector b is the inequality

$$\sum_{1 \leq i < j \leq n} b_i b_j x_{ij} \leq 0. \tag{1.5}$$

We shall refer to each inequality (1.5) as $Hyp(b)$. Note that all inequalities in (1.3), but the first, are hypermetric inequalities.

Our cutting plane algorithm goes as follow: first we find the incidence vector x^* of a cut of large weight (this is accomplished by a heuristic procedure) and then we try to prove its optimality by solving a special LP problem. If we succed in proving the optimality of x^* for problem (1.4) we stop, otherwise our algorithm produces an upper bound on the optimum value of (1.4).

The plan of the paper is as follows: Section 2 shows how we can certificate the optimality of a cut; in Section 3 we describe our cutting plane algorithm; in Section 4 we give a separation procedure for the class of hypermetric inequalities; Section 5 describes a heuristic to get 'good' feasible solutions; and finally, Section 6 reports on some computational results.

2. Certification of Optimality

Consider a graph $G = (V, E)$. The *cut cone* of G is the cone generated by all vectors in $K(G)$ and is denoted by $C(G)$. Clearly $C(G) \supset P_C(G)$; moreover a stronger relation between the two polyhedra holds. To specify such a relation, let v be a vector in $\mathbb{R}^E$. For every subset S of V, define a vector v^S in $\mathbb{R}^E$ by

$$v^S_{ij} = \begin{cases} v_{ij} & \text{if } ij \notin \delta(S) \\ -v_{ij} & \text{otherwise.} \end{cases}$$

We shall say that the vector v^S has been obtained by *switching* v with respect to the cut $\delta(S)$. Similarly, for every inequality $v^T x \leq b$ and for every cut $\delta(S)$, we shall say that the inequality $(v^S)^T x \leq b^S$, with $b^S = b - \sum_{ij \in \delta(S)} v_{ij}$ has been obtained by *switching* $v^T x \leq b$ with respect to the cut $\delta(S)$.

Barahona and Mahjoub [2] showed that for every vector v in $\mathbb{R}^E$ and for every cut $\delta(S)$, $v^T x \leq b$ defines a facet of $P_C(G)$ if and only if the inequality $(v^S)^T x \leq b^S$ defines a facet of $P_C(G)$. Furthermore, they showed that every inequality defining a facet of $P_C(G)$ can be obtained from some facet defining inequality of $C(G)$ by switching a cut [2]. Hence the complete description of $P_C(G)$ is implicitely given by the complete description of $C(G)$.

The purpose of this section is to show how such a result can be exploited algorithmically. Let x' be a solution of problem (1.1) and let S' be the subset of V corresponding to x' (x' is the incidence vector of $\delta(S')$). The following theorem yields a certification of optimality for x'.

Theorem 1. The solution x' is optimal for problem (1.1) if and only if the optimal value of the problem

$$\begin{aligned} &\max\ v^T x \\ &\text{s.t. } x \in P_C(G) \end{aligned} \tag{2.1}$$

is zero, where $v = c^{S'}$.

Proof. Write $T' = V - S'$.

(Only if part) Suppose that x' is an optimal solution of (1.1). Let $\delta(S^\star)$ be an optimal solution of problem (2.1). Set $A = S^\star \cap S'$, $B = S^\star \cap T'$, $C = S' - A$, and $D = T' - B$. The optimal value $v^\star$ of (2.1) is given by:

6.

$$\begin{aligned} v^\star &= \sum_{ij\in\delta(S^\star)} v_{ij} \\ &= \sum_{i\in A,\ \ j\in C} v_{ij} + \sum_{i\in A,\ \ j\in D} v_{ij} + \sum_{i\in B,\ \ j\in C} v_{ij} + \sum_{i\in B,\ \ j\in D} v_{ij} \\ &= \sum_{i\in A,\ \ j\in C} c_{ij} - \sum_{i\in A,\ \ j\in D} c_{ij} - \sum_{i\in B,\ \ j\in C} c_{ij} + \sum_{i\in B,\ \ j\in D} c_{ij}. \end{aligned}$$

Write $c' = \sum_{ij\in\delta(S')} c_{ij}$. We have

$$c' = \sum_{i\in A,\ \ j\in B} c_{ij} + \sum_{i\in A,\ \ j\in D} c_{ij} + \sum_{i\in B,\ \ j\in C} c_{ij} + \sum_{i\in C,\ \ j\in D} c_{ij}.$$

Now, since x' is an optimal solution of (1.1),

$$c' \geq \sum_{ij\in\delta(A\cup D)} c_{ij},$$

that is

$$c' \geq \sum_{i\in A,\ \ j\in C} c_{ij} + \sum_{i\in B,\ \ j\in D} c_{ij} + \sum_{i\in A,\ \ j\in B} c_{ij} + \sum_{i\in C,\ \ j\in D} c_{ij}.$$

Hence,

$$\sum_{i\in A,\ \ j\in C} c_{ij} + \sum_{i\in B,\ \ j\in D} c_{ij} \leq \sum_{i\in A,\ \ j\in D} c_{ij} + \sum_{i\in B,\ \ j\in C} c_{ij},$$

and so $v^\star \leq 0$. Since the zero cut is a feasible solution of (2.1), it follows that $v^\star = 0$.

(If part) Suppose now that the optimal value of problem (2.1) is zero. If x' is not optimal then there exists a cut $\delta(S)$ of G such that

$$\sum_{ij\in\delta(S)} c_{ij} > \sum_{ij\in\delta(S')} c_{ij}. \tag{2.2}$$

Set $A = S \cap S'$, $B = S \cap T'$, $C = S' - A$, and $D = T' - B$. Inequality (2.2) implies that

$$\sum_{i\in A,\ \ j\in C} c_{ij} + \sum_{i\in B,\ \ j\in D} c_{ij} > \sum_{i\in A,\ \ j\in B} c_{ij} + \sum_{i\in C,\ \ j\in D} c_{ij}. \tag{2.3}$$

Consider now the cut $\delta(A \cup D)$ of G. We have

$$\begin{aligned}\sum_{ij\in\delta(A\cup D)} v_{ij} &= \sum_{i\in A, j\in C} v_{ij} + \sum_{i\in B, j\in D} v_{ij} + \sum_{i\in A, j\in B} v_{ij} + \sum_{i\in C, j\in D} v_{ij} \\ &= \sum_{i\in A, j\in C} c_{ij} + \sum_{i\in B, j\in D} c_{ij} - \sum_{i\in A, j\in B} c_{ij} - \sum_{i\in C, j\in D} c_{ij}.\end{aligned}$$

But then (2.3) implies that

$$\sum_{ij\in\delta(A\cup D)} v_{ij} > 0,$$

contradicting the assumption that the optimal value of problem (2.1) is zero. □

The result in Theorem 1 was independently found by Barahona [1]. We shall say that the objective function $v^T x$ in (2.1) is obtained by switching the objective function $c^T x$ with respect to x'.

By Theorem 1, we can certify the optimality of a solution x' (with respect to a given objective function $c^T x$) by certifying the optimality of zero (with respect to the objective function $(c^{S'})^T x$). Note that this amounts to provide a set of linearly independent inequalities $a_i^T x \leq 0$ $(i = 1, \cdots, |E|)$ valid over $C(G)$ such that $v = \sum_i^{|E|} \lambda_i a_i$ with $\lambda_i \geq 0$ $(i = 1, \cdots, |E|)$. Hence, to prove the optimality of the origin over $P_C(G)$, we only need use the subset of the system of all the inequalities defining the cut cone $C(G)$.

Since we are allowed to work with cut cones rather than cut polytopes, it follows that checking the optimality of x' for problem (1.1) is equivalent to checking if the problem

$$\begin{aligned}&\max\ v^T x \\ &\text{s.t. } x \in C(G)\end{aligned}$$

is finite. This is an easy consequence of the fact that $C(G) \supset P_C(G)$ and that the empty cut is a feasible solution of (1.1). The above problem can be solved by a polyhedral cutting plane algorithm based on the separation of inequalities valid over $C(G)$.

The big advantage in working with cut cones $C(G)$ rather than cut polytopes $P_C(G)$ resides on the fact that the separation problem for an inequality valid over $C(G)$ is easier than the separation problem for an inequality valid over $P_C(G)$ since the former has zero as right hand side.

3. The Algorithm

Let K_n denote the complete graph with n vertices and let H_n denote the cut cone $C(K_n)$. In this section we describe a cutting plane algorithm to solve problem (1.4).

Let x^* be a 'good' solution of problem (1.4). Such a solution is produced by a heuristic algorithm (for instance, the one described in Section 5). Our goal is to prove the optimality of such a solution.

As said in the previous section, we only need show that the problem

$$\begin{aligned} &\max\ v^T x \\ &\text{s.t. } x \in H_n \end{aligned}$$

has a zero optimal solution. (Recall that $v = c^{S'}$.)

From the computation standpoint it is more convenient to work with bounded feasible sets. Hence we intersect H_n with the set of points $x \in \mathbb{R}^{\binom{n}{2}}$ satisfying the non-homogeneous triangle inequalities ($x_{ij} + x_{ik} + x_{jk} \leq 2$) and the trivial upper bound constraints $x_{ij} \leq 1$. In fact, the trivial upper bound constraints are implied by the triangle inequalities, but we used them since they can be easily handled by any LP solver.

Our cutting plane algorithm goes as follows. We first solve the problem

$$\begin{aligned} &\max\ v^T x \\ &\text{s.t. } x \in T_n \end{aligned}$$

by first solving the relaxed problem

$$\begin{aligned} &\max\ v^T x \\ &\text{s.t. } 0 \leq x \leq 1 \end{aligned}$$

and then iteratively adding every triangle inequality violated by the current optimal solution until no more triangle inequalities are violated.

Let $\overline{x}$ be the so obtained optimal solution. If $\overline{x}$ is integer then clearly we can stop: if $v^T\overline{x} = 0$ then our guess x^* was optimal for problem (1.4) and we are done; if $v^T\overline{x} > 0$ then it is easy to see that the cut obtained by taking the symmetric difference between the cut corresponding to x^* and the cut

corresponding to $\overline{x}$ is optimal for (1.4). Hence, if $\overline{x}$ is integer then we have solved our problem (1.4) to optimality.

If $\overline{x}$ is not integral, then we search for hypermetric inequalities violated by $\overline{x}$ (how this is accomplished will be shown in the next section), we use them as cutting planes, and we reoptimize.

Denote by $MC(\mathcal{L})$ the following linear program:

$$\begin{aligned} &\max\ v^T x \\ &\text{s.t. } 0 \le x \le 1 \\ &\qquad a^T x \le a_0 \text{ for all } (a, a_0) \in \mathcal{L}. \end{aligned}$$

Then a formal description of our algorithm is as follows.

ALGORITHM MAX-CUT

Step 0. Find a feasible solution x^* of (1.4) and switch the objective function $c^T x$ with respect to x^*; let $v^T x$ be the new objective function.

Step 1. Set $\mathcal{L} = \emptyset$.

Step 2. Solve the linear program $MC(\mathcal{L})$ and let $\overline{x}$ be its optimal solution.

Step 3. If $\overline{x} \in T_n$ then go to Step 5.

Step 4. Find one or more triangle inequalities violated by $\overline{x}$, add them to $\mathcal{L}$ and go to Step 2.

Step 5. If $\overline{x}$ is integral or $v^T\overline{x} < 1$, then stop; the optimal solution of (1.4) is $x^\star$. Otherwise, go to Step 6.

Step 6. Find one or more hypermetric inequalities violated by $\overline{x}$. If none is found, then stop. Otherwise, add the violated inequalities to $\mathcal{L}$ and go to Step 2.

As noted before, when we stop at Step 5 we have solved our problem (1.4) to optimality. The situation is different when we stop at Step 6. In this case, denoting by $v^\star$ the optimum value of $MC(\mathcal{L})$, the algorithm produces an upper bound on the optimum of (1.4), namely $c^T x^* + v^*$. What we can

do, in this case, is either use techniques of enumerative type like branch and cut [10] or try to exploit the current fractional optimal solution to improve the initial guess x^*. (In the latter case, if a better feasible solution is found, we switch the objective function with respect to the new solution and restart from Step 2.) However, we have not developed this part of the algorithm yet and we leave it for future work. Consequently, when the algorithm stops at Step 6 it only produces a feasible solution $x^\star$ along with an upper bound on the gap between its value ($c^T x^\star$) and the optimal one.

To complete the description of our algorithm, it remains to show how we find an initial solution x^* of problem (1.4) in Step 0, and how we find hypermetric inequalities violated by $\overline{x}$ in Step 6. In the following section, we shall first show how we solve the separation problem for the hypermetric inequalities, and we postpone the description of an algorithm to find x^* to Section 5.

4. A Separation Heuristic for the Hypermetric Inequalities

In this section we assume to have an optimal fractional solution $\overline{x}$ of problem $MC(\mathcal{L})$.

Our separation problem amounts to

$$\begin{aligned} &\textit{find integers} \quad b_1, \cdots, b_n \\ &\qquad s.t. \sum_{1 \le i < j \le n} b_i b_j \overline{x}_{ij} > 0, \quad \sum_{i=1}^{n} b_i = 1. \end{aligned} \tag{4.1}$$

We shall show that problem (4.1) can be reduced to the problem of finding cuts whose weights satisfy a certain property in a special complete graph.

To do that, we first eliminate in (4.1) the constraint $\sum_{i=1}^{n} b_i = 1$ by setting $b_n = 1 - \sum_{i=1}^{n-1} b_i$. Let Q denote the $(n-1) \times (n-1)$ symmetric matrix whose ij-th element is defined by

$$q_{ij} = \begin{cases} \overline{x}_{ij} - \overline{x}_{in} - \overline{x}_{jn} & \text{if } i \neq j \\ -2\overline{x}_{jn} & \text{otherwise,} \end{cases}$$

and let d denote the vector whose i-th element is defined by

$$d_i = 2\overline{x}_{in}, \quad i = 1, \cdots, n-1.$$

Defining $b = [b_1, b_2, \cdots, b_{n-1}]^T$, we can write

$$\sum_{1 \le i < j \le n} b_i b_j \overline{x}_{ij} = b^T Q b + d^T b,$$

and so problem (4.1) amounts to

find an integral vector b such that $b^T Q b + d^T b > 0$.

Now, let v be a given vector in $\mathbb{R}^{n-1}$. Clearly, if v is integral and $v^T Q v + d^T v > 0$, then the inequality $Hyp(b_1, \cdots, b_n)$, with $b_i = v_i$ for every $i = 1, \cdots, n-1$ and $b_n = 1 - \sum_{i=1}^{n-1} b_i$, is an hypermetric inequality violated by the point $\overline{x}$.

Assume this is not the case and set

$$w_i = \lfloor v_i \rfloor - \alpha_i + \beta_i, \quad i = 1, \cdots, n-1, \tag{4.2}$$

where each α_i and β_i is a $0-1$ variable. Since by construction w is an integral vector, our goal is then to choose a *bias* vector v and to assign values to the variables α_i and β_i $(i = 1, \cdots, n-1)$ in such a way that $w^T Q w + d^T w > 0$.

Now, it is a routine but tedious matter to verify that the function $b^T Q b + d^T b$, evaluated at the point $w = [w_1, w_2, \cdots, w_{n-1}]^T$, can be written as a quadratic $0-1$ function, that is

$$w^T Q w + d^T w = x^T B x + a^T x + C,$$

where

$$x = [\alpha_1, \cdots, \alpha_{n-1}, \beta_1, \cdots, \beta_{n-1}]^T,$$

B is a $(n-1) \times (n-1)$ matrix, a is a $n-1$ vector, and C is a constant.

Now, consider the problem

$$\begin{aligned} &\max\ x^T B x + a^T x \\ &\text{s.t. } x \in \{0,1\}^{2n-2}. \end{aligned} \tag{4.3}$$

Clearly, each solution of (4.3) that has value greater than $-C$ yields an hypermetric inequality violated by the point $\overline{x}$. We call such a solution a

generating solution. Since problem (4.3) is an unconstrained $0-1$ quadratic problem, it can be solved as a max-cut problem on the graph $H = K_{2(n-1)+1}$ with suitably defined weights (see [5]). Moreover, since we are not interested in optimal but rather in generating solutions of (4.3), it follows that finding generating solutions of (4.3) is equivalent to finding cuts of H having weight greater than $-C$ (*generating cuts*).

Hence, we have shown that to solve problem (4.1), we can use any good heuristic for the solution of max-cut problems on complete graphs. (This fact motivates our choice of postponing the description of our heuristic to the next section.)

Now, since we are not interested in optimal solutions of (4.3), when we run our heuristic we collect not only the best but all the generating cuts found (if any). In this way, we have the possibility of generating several hypermetric inequalities at the same time. On the other hand, it may happen that too many of them are generated and this may slow down the LP solver. Hence, we rank all the generated hypermetric inequalities according to their violation divided by the norm of the coefficients vector, and we take the first k according to this ranking, where k is a parameter that we have found sperimentally.

Now it remains to show how we choose the vector v.

A first choice for the bias vector v is to take the zero vector; in this case, any generating cut will yield an hypermetric inequality $Hyp(b_1, \cdots, b_n)$ such that $b_1, \cdots, b_{n-1}$ take values in the range $[-1, 0, 1]$. The class of these hypermetric inequalities is a very interesting one at least for the following three reasons:

(1) it contains all the clique inequalities of Barahona and Majoub [2] and their liftings with zero coefficients (an inequality of this class is actually a clique inequality when $b_n = \pm 1$);
(2) from our computational experience, we found them very effective;
(3) usually they define facets of H_n. Here are some conditions under which this is true (see Theorem 3.12 in [8]):
- $b_n < 0$, there are at least three b_i that have value 1, and at least two b_i that have value ≤ 0;

- $b_n > 0$, there are at least two b_i that have value 1, and at least two b_i that have value ≤ 0.

Now, to allow the coefficients $b_1, \cdots, b_{n-1}$ to take also values different from $-1, 0, 1$, we choose the bias vector v in a different way. Since we noticed that the hypermetric inequalities that are really effective as cutting planes have coefficients with small absolute value, we restrict our separation procedure to identify hypermetric inequalities whose first $n-1$ coefficients take values in the range $[-2, -1, 0, 1, 2]$. Note that to do so, the components of the bias vector v have to take values in the range $[-1, 0, 1]$.

To compute the vector v we use the eigenvalues of the matrix Q. Let λ be the vector of all $n-1$ eigenvalues of Q and let U be the matrix whose columns are the eigenvectors corresponding to the components of λ. If all components of λ are different from zero, then the problem

$$\begin{aligned} &\max\ b^T Q b + d^T b \\ &\text{s.t. } b \in \mathbb{R}^{n-1} \end{aligned} \tag{4.4}$$

has a stationary point

$$v_1 = -\frac{1}{2} Q^{-1} d = -\frac{1}{2} U L^{-1} U^T d,$$

where $L = diag(\lambda)$. Let λ_{n-1} denote the maximum eigenvalue of the matrix Q. We distinguish among two cases:

(a) $\lambda_{n-1} \leq 0$,

(b) $\lambda_{n-1} > 0$.

In case (a), if $v_1^T Q v_1 + d^T v_1 \leq 0$ then problem (4.4) has zero as optimal value, and so problem (4.1) has no solutions: the point $\overline{x}$ violates no hypermetric inequalities; otherwise we choose $v = v_1$.

Assume now that we are in case (b). Let u denote the last column of U (u is the eigenvector corresponding to the maximum eigenvalue of Q); set $y = U^T b$ and $t = U^T d$. We have

$$\begin{aligned} b^T Q b + d^T b &= y^T L y + t^T y \\ &= \lambda_1 (y_1)^2 + t_1 y_1 + \cdots + \lambda_{n-1} (y_{n-1})^2 + t_{n-1} y_{n-1}. \end{aligned}$$

Note that $y^T Ly + t^T y$ attains a strictly positive value for

$$y = [0, 0, \cdots, 0, e]^T$$

for every $e > 0$ if $t_{n-1} > 0$, and for every $e < 0$ if $t_{n-1} < 0$. Let y be a vector in $\mathbb{R}^{n-1}$ defined by $y = u$ if $t_{n-1} > 0$ and $y = -u$ otherwise. Our choice for v is then a vector whose i-th component takes the value 1 if y_i is a relatively 'big' positive number, -1 if y_i is a relatively 'big' negative number, and 0 otherwise.

Once we have found an hypermetric inequality violated by $\overline{x}$, there is a simple way to produce another one. Consider an hypermetric inequality $Hyp(b_1, \cdots, b_n)$ and let b' be the vector defined by

$$b'_i = \begin{cases} -b_i & \text{if } i = 1, \cdots, n-1 \\ 1 - \sum_{i=1}^{n-1} b'_i & \text{if } i = n. \end{cases}$$

Clearly, the inequality

$$\sum_{1 \le i < j \le n} b'_i b'_j x_{ij} \le 0,$$

is an hypermetric inequality different from $Hyp(b_1, \cdots, b_n)$ because $b'_n = 1 - \sum_{i=1}^{n-1} b'_i = 1 + \sum_{i=1}^{n-1} b_1 = 2 - b_n$. Furthermore, if the fractional point $\overline{x}$ violates $Hyp(b_1, \cdots, b_n)$, then it may also violate $Hyp(b'_1, \cdots, b'_n)$. Hence, every time the heuristic returns a generating cut whose corresponding hypermetric inequality is $Hyp(b_1, \cdots, b_n)$, we have another hypermetric inequality that, if it is violated, it can be added to the cutting plane list.

Before closing this section, we want to remark that the separation procedure for the hypermetric inequalities can also be used in a cutting plane algorithm to solve max-cut problems on general graphs. To see this, consider a graph $G = (V, E)$ and assume that G has some cliques K_i of size at least five. Then every hypermetric inequality $Hyp(b)$ valid over $P_C(K_i)$ can be lifted with zero coefficients to an inequality $v^T x \le 0$ valid over $P_C(G)$; moreover, if $Hyp(b)$ defines a facet of $P_C(K_i)$ then $v^T x \le 0$ also defines a facet of $P_C(G)$ (see [6], [7]).

5. Computing Lower Bounds

In this section, we describe a heuristic procedure for the max-cut problem on a general graph.

Let $G = (V, E)$ be a graph having weights assigned to its edges and let S be a subset of V. We shall say that the cut $\delta(S)$ is *1-opt* (or optimum with respect to *1-exchange*) if its weight is greater than the weight of every cut $\delta(S - \{i\})$ $(i \in S)$ and of every cut $\delta(S \cup \{j\}$ $(j \notin S)$ of G. We shall say that $\delta(S)$ is *2-opt* (or optimum with respect to *2-exchange*) if its weight is greater than the weight of every cut $\delta((S - \{i\}) \cup \{j\})$ $(i \in S,\, j \notin S)$ of G. A cut $\delta(S)$ which is both 1-opt and 2-opt is called a $(1-2)$-*opt* cut. Finally, to *postoptimize* a cut it means to perform repeated 1-exchange and 2-exchange on this cut until a $(1-2)$-opt cut is obtained.

Our heuristic is composed by three different procedures, all having as input a pair (G, c), that is a graph G along with a vector c of weights assigned to its edges.

The first procedure simply produces a random cut which is then postoptimized.

The second procedure orders the vertices of G in a random way; let $v_1, v_2, \cdots, v_n$ be such an ordering. For $i = 1, \cdots, n$, we shall denote by G_i the subgraph of G induced by the subset of vertices $\{v_1, v_2, \cdots, v_i\}$. Each edge of G_i has the weight of the corresponding edge of G.

For every $i = 1, \cdots, n$ a cut of G_i is denoted by $\delta(S_i)$. Clearly, the empty cut is a $(1-2)$-opt cut of G_1. Let $\delta(S_i)$ be a $(1-2)$-opt cut of G_i. Then to find a cut of G_{i+1} which is $(1-2)$-opt, we only need postoptimize the cut $\delta(S_i \cup \{v_{i+1}\})$. Clearly, the final cut $\delta(S_n)$ (after it has been postoptimized) is a $(1-2)$-opt cut of G.

Let W be the weight of $\delta(S_n)$. If $W = 0$ then we stop. Otherwise, we switch the objective function $c^T x$ with respect to such a cut; let $(c')^T x$ denote the new objective function so obtained $(c' = c^{S_n})$. Run the procedure for the pair (G, c') and let W' be the weight of the $(1-2)$-opt cut $\delta(S'_n)$ so obtained. It is easy to see that the cut $\delta((S - S') \cup (S' - S))$ has weight $W + W'$. Hence, if W' is positive we obtain a new cut of G having weight greater than W.

Hence, every time the procedure finds a cut of positive weight, it switches

the current objective function with respect to such a cut and it restarts.

Finally, the last procedure has as input along with (G, c) an integer $M < n$. We compute the eigenvector y corresponding to the maximum eigenvalue of the matrix L associated with the weights of G ($L_{ij} = L_{ji} = c_{ij}$). We order the components of y in non decreasing order: let $y_1 \leq y_2 \leq \cdots y_n$ be such an ordering and let $v_1, v_2, \cdots, v_n$ be the corresponding ordering of the vertices of G. We build n different cuts $\delta(S_1), \delta(S_2), \cdots, \delta(S_n)$ of G in such a way that the corresponding weights W_i ($i = 1, \cdots, n$) verify $W_1 \leq W_2 \leq \cdots \leq W_n$. This is done as follows. Set $S_0 = \emptyset$. Assume to have built the cut $\delta(S_{i-1})$; without loss of generality, we can suppose that $v_i \in S_{i-1}$. If the weight of the cut $\delta(S_{i-1} - \{v_i\})$ is greater than W_{i-1} then $\delta(S_i) = \delta(S_{i-1} - \{v_i\})$; otherwise, $\delta(S_i) = \delta(S_{i-1})$.

We then check wether $W_n = 0$; if it is so then the procedure returns the empty-cut. Otherwise, let

$$k = \max\{i : W_{i-1} < W_i\}.$$

If $2 + \frac{M-1}{2} \leq k \leq n - 1 - \frac{M-1}{2}$, then we set $p = k - \frac{M-1}{2} - 1$ and $q = k + \frac{M-1}{2} + 1$. If $k - \frac{M-1}{2} - 1 \leq 0$, then we set $p = 0$ and $q = M + 1$. If $k > n - 1 - \frac{M-1}{2}$, then we set $p = n - M$ and $q = n + 1$. Once p and q have been computed, 2^M different cuts $\delta(S)$ are built, where S is the union of $\{v_1, \cdots, v_p\}$ and of any of the 2^M subsets of $\{v_{p+1}, \cdots, v_{p+M}\}$. Finally, we postoptimize the best of these cuts.

6. Computational Experiments

In this section we report our findings concerning the practical performance of our cutting plane algorithm for solving max-cut problems on complete graphs.

Unfortunately, we could find only few papers reporting computational results on the solution of max-cut problems on complete graphs or of the associated quadratic $0-1$ problems (UBQ) with fully dense matrix. Actually, the only papers we could find were about quadratic $0-1$ problems (see [3], [4], and [11]). The problems solved in these three papers are of two types, namely *Williams*-type and *Carter*-type. Disappointingly though, when we converted these problems into max-cut problems, our algorithm could solve practically all instances we generated without involving the separation procedure of the hypermetric inequalities at all: the triangle inequalities were usually sufficient to prove the optimality of the cut produced by the heuristic. This is in agreement with what is remarked in [3]. For this reason, we do not report on the experiments we have done with these problems.

Hence, we tried to build instances of max-cut problems for which the only triangle inequalities were not sufficient to prove optimality. For this purpose, we followed two different approaches.

In the first one, we generated max-cut problems by randomly picking a certain *percentage* $P\%$ of edges in a complete graph and by assigning a weight W to these edges and zero to the others. The results obtained for these type of problems are shown in Tables 1-2-3-4 for $W = 1$, and in Tables 5-6 for W randomly chosen in $\{0, 100\}$. Most of these problems turned out to require the generation of hypermetric inequalities. Notice that these max-cut problems are 'nasty' for our algorithm since they treat graphs that actually have density $P\%$ as complete; for instance a complete graph having $P = 20$ is sparse and so it has many cycles whose corresponding cycle inequalities could be used as cutting planes.

The second way we followed was to study the behavior of our algorithm when the objective function is a facet-defining inequality. Now, the largest cut polytope of a complete graph that is known nowadays is $P_C(K_7)$ ([6], [9]). Since H_7 has precisely 38780 facets, we just picked the 36 representatives of them, that is the ones from which all others can be obtained by permuting

the vertices. They come into four groups: hypermetric inequalities, *cycle* inequalities, *parachute* inequalities, and *Grishouhine* inequalities. With only few exceptions, all these problems needed to call the separation procedure of Section 4 (see Tables 7-8-9-10).

The representatives of the first group are:

$$Hyp(1,1,-1,0,0,0,0); \qquad (H1)$$
$$Hyp(1,1,1,-1,-1,0,0); \qquad (H2)$$
$$Hyp(1,1,1,1,-1,-1,-1); \qquad (H3)$$
$$Hyp(2,1,1,-1,-1,-1,0), \qquad (H4)$$
$$Hyp(-2,1,1,1,1,-1,0); \qquad (H5)$$
$$Hyp(2,2,1,-1,-1,-1,-1), \qquad (H6)$$
$$Hyp(-2,2,1,1,1,-1,-1), \qquad (H7)$$
$$Hyp(-2,-2,1,1,1,1,1); \qquad (H8)$$
$$Hyp(3,1,1,-1,-1,-1,-1), \qquad (H9)$$
$$Hyp(-3,1,1,1,1,1,-1); \qquad (H10)$$

To list the representatives of all other inequalities, let

$$u = (0,1,1,0,-1,-1;0,1,1,-1,-1;0,1,-1,-1;0,-1,-1;-1,-1;1)^T;$$
$$v = (1,2,2,-2,-2,-2;1,2,-2,-2,-2;0,-1,-1,-1;-1,-1,-1;1,1;1)^T;$$
$$w = (5,5,-3,-3,-3,-3;3,-2,-2,-2,-2;-2,-2,-2,-2;1,1,1;1,1;1)^T;$$
$$p = (0,-1,-1,-1,-1,0;1,0,-1,-1,-1;1,0,0,-1;1,0,-1;1,0;1)^T;$$
$$g = (1,1,1,-2,-1,0;1,1,-2,0,-1;1,-2,-1,0;-2,0,-1;1,1;-1)^T,$$

where the components of the five vectors are in a increasing lexicographic order. Recall that for every vector v and for every cut $\delta(S)$, v^S denotes the vector obtained by switching v with respect to the cut $\delta(S)$.

Then, representatives of the second group are:

$$u^T x \leq 0, \tag{C1}$$
$$v^T x \leq 0, \tag{C2}$$
$$w^T x \leq 0, \tag{C3}$$
$$(u^{\{1\}})^T x \leq 0, \tag{C4}$$
$$(u^{\{1,2\}})^T x \leq 0, \tag{C5}$$
$$(u^{\{1,2,6\}})^T x \leq 0, \tag{C6}$$
$$(v^{\{1\}})^T x \leq 0, \tag{C7}$$
$$(v^{\{3\}})^T x \leq 0, \tag{C8}$$
$$(v^{\{1,5\}})^T x \leq 0, \tag{C9}$$
$$(v^{\{3,4\}})^T x \leq 0, \tag{C10}$$
$$(v^{\{1,4,5\}})^T x \leq 0, \tag{C11}$$
$$(v^{\{3,4,5\}})^T x \leq 0, \tag{C12}$$
$$(w^{\{2\}})^T x \leq 0, \tag{C13}$$
$$(w^{\{2,4\}})^T x \leq 0, \tag{C14}$$
$$(w^{\{1,4\}})^T x \leq 0, \tag{C15}$$
$$(w^{\{1,4,5\}})^T x \leq 0; \tag{C16}$$

representatives of the third group are:

$$p^T x \leq 0, \tag{P1}$$
$$(p^{\{3,7\}})^T x \leq 0, \tag{P2}$$
$$(p^{\{1,3,6\}})^T x \leq 0; \tag{P3}$$

representatives of the fourth group are:

$$g^T x \leq 0, \tag{G1}$$
$$(g^{\{1\}})^T x \leq 0, \tag{G2}$$

$$(g^{\{1,6\}})^T x \leq 0, \tag{$G3$}$$

$$(g^{\{1,6,7\}})^T x \leq 0, \tag{$G4$}$$

$$(g^{\{1,3,5\}})^T x \leq 0, \tag{$G5$}$$

$$(g^{\{1,3,6\}})^T x \leq 0, \tag{$G6$}$$

$$(g^{\{1,2,5\}})^T x \leq 0. \tag{$G7$}$$

In the following we give the results of our experiments in several tables using the following abbreviations:

n : number of vertices;

P : the percentage defined before;

SEP: number of calls of the separation procedure for the hypermetric inequalities;

PIV: number of pivots;

TRS: optimal solution found over the triangle polytope;

GAP: optimal solution of $MC(\mathcal{L})$;

TLB: CPU time in seconds spent to produce a lower bound;

TTR: CPU time in seconds to compute TRS;

TLP: CPU time in seconds spent in the LP solver;

TT: total CPU time in seconds spent by the whole algorithm.

To keep the number of constraints of each linear program small, we decided to limit the number of triangle inequalities added each time we perform Step 3 of Algorithm MAX-CUT, by $\min\{\frac{n^2}{3}, 300\}$.

As can be seen from all tables, most of the time is spent in the LP solver. Indeed, one needs at first an optimal basis on the triangle polytope. But, while the initial basis of the LP is made only of trivial inequalities ($0 \leq x_{ij} \leq 1$), which are never facet-inducing in a complete graph, the optimal one is made exclusively (ignoring exceptional degenerated cases) of triangle inequalities.

Table 1.

Problem	n	P	*PIV*	*SEP*	*TRS*	*GAP*
10SV10	10	10	0	0	0*	0*
10SV20	10	20	24	0	0*	0*
10SV30	10	30	47	0	0*	0*
10SV40	10	40	35	0	0*	0*
10SV50	10	50	28	0	0*	0*
10SV60	10	60	19	0	0*	0*
10SV70	10	70	122	0	.66	.66
10SV80	10	80	97	0	.66	.66
10SV90	10	90	127	1	1.0	.25
15SV10	15	10	0	0	0*	0*
15SV20	15	20	122	1	1	1*
15SV30	15	30	255	1	1	1*
15SV40	15	40	231	0	0*	0*
15SV50	15	50	279	0	.50	.50
15SV60	15	60	195	0	.50	.50
15SV70	15	70	1089	2	2.66	.89
15SV80	15	80	1094	2	4.33	.95
15SV90	15	90	1189	2	8.00	1*

* integer solution

This requires that we go from an initial basis to a final one in a number of pivots which is at least m, the number of edges of K_n ($m = \binom{n}{2}$). In fact, the number of pivots is much greater than m. Since our goal was to show how the hypermetric inequalities could be effective in a cutting plane algorithm, we were unwilling to spend much CPU time to get more results on these type of problems, and so we decided to keep the size of the graphs small.

Table 2.

Problem	n	P	TLB	TLP	TTR	TT
10SV10	10	10	.05	.03	.33	.33
10SV20	10	20	.08	.28	.77	.77
10SV30	10	30	.05	.72	1.28	1.28
10SV40	10	40	.06	.47	.98	.98
10SV50	10	50	.07	.27	.71	.71
10SV60	10	60	.07	.22	.63	.63
10SV70	10	70	.06	2.82	3.45	3.45
10SV80	10	80	.05	1.86	2.41	2.41
10SV90	10	90	.05	3.08	2.97	4.47
15SV10	15	10	.12	.05	.66	.66
15SV20	15	20	.11	3.58	5.28	7.28
15SV30	15	30	.13	9.06	10.70	12.74
15SV40	15	40	.12	8.20	9.74	9.74
15SV50	15	50	.14	9.74	11.13	11.13
15SV60	15	60	.16	6.18	7.33	7.33
15SV70	15	70	.16	68.21	35.20	76.62
15SV80	15	80	.12	77.68	22.31	86.06
15SV90	15	90	.13	86.91	23.83	95.07

However, we succeed in showing the effectiveness of the hypermetric inequalities in a cutting plane environment: among the 98 cases analyzed, in only one case, namely (C13) in Table 8, we could not prove the optimality of the cut found by the heuristic.

The computational test showed that our heuristic too is quite effective. With only few exceptions, it was able to find an optimal cut. When the heuristic failed, the hypermetric inequalities that were added could solve the problem to optimality, still with the exception of problem ($C13$).

Table 3.

Problem	*n*	*P*	*PIV*	*SEP*	*TRS*	*GAP*
20SV10	20	10	116	1	1.0	1*
20SV20	20	20	135	0	0*	0*
20SV30	20	30	1023	0	.50	.50
20SV40	20	40	1738	0	.77	.77
20SV50	20	50	1300	0	.38	.38
20SV60	20	60	1672	0	.30	.30
20SV70	20	70	6228	3	6.0	.90
20SV80	20	80	3637	3	8.3	.83
20SV90	20	90	6382	2	14.0	.95

* integer solution

Table 4.

Problem	*n*	*P*	*TLB*	*TLP*	*TTR*	*TT*
20SV10	20	10	.18	3.95	6.03	9.53
20SV20	20	20	.24	4.26	6.32	6.32
20SV30	20	30	.33	72.43	76.08	76.08
20SV40	20	40	.22	120.10	126.11	126.11
20SV50	20	50	.30	84.27	88.40	88.40
20SV60	20	60	.20	110.27	114.78	114.78
20SV70	20	70	.23	761.85	153.56	799.41
20SV80	20	80	.27	403.50	113.01	420.73
20SV90	20	90	.27	832.37	162.03	852.16

Table 5.

Problem	*n*	*SEP*	*PIV*	*TRS*	*GAP*
10SV	10	91	0	0*	0*
15SV	15	563	0	0*	0*
20SV	20	3002	1	165	165*
25SV	25	15558	9	298.33	64*

* integer solution

Table 6.

Problem	*n*	*TLB*	*TLP*	*TTR*	*TT*
10SV	10	.07	1.69	.34	2.37
15SV	15	.15	26.13	27.95	27.95
20SV	20	.22	260.41	265.77	269.48
25SV	25	.45	3717.14	815.54	3892.66

Table 7.

Problem	*PIV*	*SEP*	*TRS*	*GAP*	*TLB*	*TLP*	*TTR*	*TT*
(H1)	1	0	0*	0*	.02	.04	.33	.33
(H2)	20	0	.66	.66	.01	.23	.52	.52
(H3)	49	1	2.00	0*	.04	.70	.73	1.52
(H4)	31	1	1.33	.66	.03	.42	.65	1.26
(H5)	46	2	1.33	.66	.03	.78	.66	2.47
(H6)	29	1	2.66	0*	.04	.39	.60	1.21
(H7)	32	1	2.66	0*	.03	.45	.70	1.25
(H8)	28	1	2.66	0*	.02	.39	.64	1.14
(H9)	44	1	2.00	.66	.02	.50	.54	1.26
(H10)	45	2	2.00	.00	.01	.70	.70	2.28

* integer solution

Table 8.

Problem	*PIV*	*SEP*	*TRS*	*GAP*	*TLB*	*TLP*	*TTR*	*TT*
(C1)	25	0	.80	.80	.03	.33	.68	.68
(C2)	36	1	2.00	.80	.05	.60	.67	1.41
(C3)	78	5	3.33	.92	.02	1.85	.65	5.94
(C4)	32	0	.80	.80	.03	.45	.77	.77
(C5)	29	0	.80	.80	.02	.34	.61	.61
(C6)	27	0	.80	.80	.04	.34	.67	.67
(C7)	40	1	2.00	.750	.02	.57	.63	1.42
(C8)	58	3	2.00	.77	.02	1.00	.59	3.31
(C9)	44	1	2.00	.83	.02	.57	.62	1.33
(C10)	48	3	2.00	.85	.03	.75	.52	2.72
(C11)	51	2	2.00	.88	.04	.89	.69	2.47
(C12)	49	2	2.00	.88	.03	.77	.65	2.36
(C13)	57	4	3.33	1.15∗	.02	1.06	.54	3.75
(C14)	55	2	3.33	.88	.02	.79	.62	2.02
(C15)	57	4	3.33	.94	.02	.72	.72	3.95
(C16)	64	4	3.33	.82	.04	1.18	.62	3.97

* optimal solution not found

Table 9.

Problem	*PIV*	*SEP*	*TRS*	*GAP*	*TLB*	*TLP*	*TTR*	*TT*
(P1)	27	0	.80	.80	.02	.35	.67	.67
(P2)	33	0	.80	.80	.03	.40	.75	.75
(P3)	29	0	.80	.80	.02	.36	.69	.69

Table 10.

Problem	*PIV*	*SEP*	*TRS*	*GAP*	*TLB*	*TLP*	*TTR*	*TT*
(G1)	39	2	1.33	.88	.02	.62	.70	2.27
(G2)	43	2	1.33	.82	.04	1.33	.69	2.27
(G3)	41	1	1.33	.88	.02	.64	.64	1.32
(G4)	43	2	1.33	.88	.02	.70	.70	2.81
(G5)	37	2	1.33	.93	.02	.68	.68	2.18
(G6)	28	1	1.33	.85	.02	.65	.65	1.19
(G7)	47	2	1.33	.83	.03	.58	.58	2.16

The algorithm described in this paper has been implemented in standard FORTRAN 77 and the computational tests have been done on a VAX 6310 machine at the Istituto di Analisi dei Sistemi ed Informatica del Consiglio Nazionale delle Ricerche in Rome. For the LP solver we used XMP library and for the eigenvalue computations the NAG library. It can be reasonably expected that a more efficient LP solver code would decrease the total running time of the algorithm substantially.

References

[1] F. Barahona (personal communication, 1991).

[2] F. Barahona and and A.R. Mahjoub: On the cut polytope, *Mathematical Programming* 36 (1986) 157-173.

[3] F. Barahona, M. Jünger, and G. Reinelt: Experiments in quadratic $0-1$ Programming, *Mathematical Programming* 44 (1989) 127-137.

[4] M.W. Carter: The indefinite zero-one quadratic problem, *Discrete Applied Mathematics* 7 (1984) 23-44.

[5] C. De Simone: The Cut Polytope and the Boolean Quadric Polytope, *Discrete Mathematics* 79 (1989) 71-75.

[6] C. De Simone, M. Deza, and M. Laurent: Collapsing and Lifting for the Cut Cone, Proceedings of the *Second Japan Conference on Graph Theory and Combinatorics*, Hakone, Japan, August 18-22, 1990.

[7] C. De Simone: The Max Cut Problem, Ph.D. Thesis, Rutgers University, (1991).

[8] M. Deza and M. Laurent: Facets for the cut cone I, *Mathematical Programming* 56 (1992) 121-160.

[9] V.P. Grishukhin: All facets of the Hamming cone for $n = 7$ are known, *European Journal of Combinatorics* 11 (1990) 115-117.

[10] M. Padberg and G. Rinaldi: Optimization of a 532-city symmetric travelling salesman problem by branch and cut, *Operations Research Letters* 6 (1986) 1-7.

[11] A.C. Williams: Quadratic $0-1$ programming using the roof dual with computational results, RUTCOR Research Report 8-85, The State Univ. of New Jersey (1985).