A semidefinite programming based polyhedral cut and price approach for the maxcut problem[1] [2]

Kartik Krishnan
Harrelson Hall 235
Department of Mathematics
Campus Box 8205
North Carolina State University
Raleigh, NC 27695-8205
kksivara@ncsu.edu
http://www4.ncsu.edu/~kksivara

John E. Mitchell
Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180
mitchj@rpi.edu
http://www.rpi.edu/~mitchj

Draft of September 7, 2005

Abstract

We investigate solution of the maximum cut problem using a polyhedral cut and price approach. The dual of the well-known SDP relaxation of maxcut is formulated as a semi-infinite linear programming problem, which is solved within an interior point cutting plane algorithm in a dual setting; this constitutes the pricing (column generation) phase of the algorithm. Cutting planes based on the polyhedral theory of the maxcut problem are then added to the primal problem in order to improve the SDP relaxation; this is the cutting phase of the algorithm. We provide computational results, and compare these results with a standard SDP cutting plane scheme.

Keywords: Semidefinite programming, column generation, cutting plane methods, combinatorial optimization

[1]Research supported in part by NSF grant numbers CCR–9901822 and DMS–0317323
[2]Work done as part of the first authors Ph.D. dissertation at RPI

1 Introduction

Let $G = (V, E)$ denote an edge weighted undirected graph without loops or multiple edges. Let $V = \{1, \ldots, n\}$, $E \subset \{\{i, j\} : 1 \leq i < j \leq n\}$, and $w \in I\!R^{|E|}$, with $\{i, j\}$ the edge with endpoints i and j, and weight w_{ij}. We assume that $m = |E|$. For $S \subseteq V$, the set of edges $\{i, j\} \in E$ with one endpoint in S and the other in $V \backslash S$ form the *cut* denoted by $\delta(S)$. We define the weight of the cut as $w(\delta(S)) = \sum_{\{i,j\} \in \delta(S)} w_{ij}$. The *maximum cut problem*, denoted as (MC), is the problem of finding a cut for which the total weight is maximum. (MC) can be formulated as

$$\max\{w(\delta(S)) | S \subseteq V\} \qquad (MC) \tag{1}$$

The problem finds numerous applications in, for example, the layout of electronic circuitry (Chang & Du [7] and Pinter [40]), and state problems in statistical physics (Barahona et al. [3]). It is also a canonical problem from which one can develop algorithms for other combinatorial optimization problems such as min-bisection, graph partitioning etc.

The maximum cut problem is one of the original NP complete problems (Karp [22]). However, there are several classes of graphs for which the maximum cut problem can be solved in polynomial time; for instance planar graphs, graphs with no K_5 minor etc. A good survey on the maxcut problem appears in Poljak and Tuza [41].

We can model the maxcut problem as an integer program as follows: Consider a binary variable associated with each edge of the graph. The value of the variable indicates whether the two endpoints of the edge are on the same side of the cut or on opposite sides. Taken together, these binary variables define a binary vector, which represents a feasible solution to the maxcut problem. The convex hull of all these candidate solutions forms a polytope known as the maxcut polytope, whose complete description is unknown. The maxcut problem amounts to minimizing a linear objective function over this polytope. One can also regard these variables as the entries of a matrix which has a row and a column for each vertex, and this matrix should be rank one (and therefore positive semidefinite) for the values of the variables to be consistent. This gives rise to a non-polyhedral (SDP) formulation of the maxcut problem where one is now optimizing over this matrix variable. These non-polyhedral formulations can be solved efficiently and in polynomial time using interior point methods. In an LP (SDP) cutting plane approach, one begins with an initial polyhedral (non-polyhedral) description that contains the maxcut polytope, and these descriptions are progressively refined using cutting planes. The procedure is repeated until we have an optimal solution to the maxcut problem.

Our aim in this paper is to solve the maxcut problem to optimality using a polyhedral cut and price approach and heuristics. Although, our approach is entirely polyhedral, it closely mimics an SDP cutting plane approach for the maxcut problem.

The paper is organized in a manner that we have all the details at hand to describe our cutting plane approach, which appears in §4. An outline of the paper is as follows: §2 deals with a linear programming (LP) formulation while §3 deals with a semidefinite programming (SDP) formulation of the maxcut problem. In §4 we motivate and develop our polyhedral cut and price approach for the maxcut problem; this is based on an SDP formulation of the maxcut problem presented in §3. The details of the algorithm are given in §5. We present some computational results in §6 and our conclusions in §7.

1.1 Notation

A quick word on notation: we represent vectors and matrices, by lower and upper case letters respectively. For instance d_{ij} denotes the jth entry of the vector d_i in a collection, whereas X_{ij} denotes the entry in the ith row and jth column of matrix X. The requirement that a matrix X be symmetric and positive semidefinite (psd) is expressed $X \succeq 0$. We also use MATLAB-like notation frequently in this paper. For instance $y(n+1:m)$ will denote components $n+1$ to m of vector y, while $A(n+1:m, 1:p)$ will denote the sub-matrix obtained by considering rows $n+1$ to m, and the first p columns of matrix A. Finally $d_j.^2$ refers to a *row* vector obtained by squaring all the components of d_j.

2 Linear programming formulations of the maxcut problem

Consider a variable x_{ij} for each edge $\{i,j\}$ in the graph. Let x_{ij} assume the value 1 if edge $\{i,j\}$ is in the cut, and 0 otherwise. The maxcut problem can then be modelled as the following integer programming problem.

$$\begin{array}{llll} \max & \sum_{i=1}^{n} \sum_{i<j, \{i,j\} \in E} w_{ij} x_{ij} & & \\ \text{subject to} & x & \text{is the incidence vector of a cut,} \end{array} \tag{2}$$

where n is the number of vertices in the graph.

Let $\mathrm{CUT}(G)$ denote the convex hull of the incidence vectors of cuts. Since maximizing a linear function over a set of points is equivalent to maximizing it over the convex hull of this set of points, we can also write (2) as the following linear program

$$\begin{array}{ll} \max & c^T x \\ \text{s.t.} & x \in \mathrm{CUT}(G). \end{array} \tag{3}$$

Here $c = \{w_{ij}\},\ x = \{x_{ij}\} \in I\!R^m$, where m is the number of edges in the graph. We usually index c and x with the subscript e to relate them to the edges in the graph. However from time to time we drop the subscript e as appropriate. If we had an oracle capable of solving the separation problem with respect to this polytope in polynomial time, we could use this oracle in conjunction with the ellipsoid algorithm to solve the maxcut problem in polynomial time (Grötschel et al [15]). Unfortunately we do not have polynomial time separation oracles for some of the inequalities that describe the maxcut polytope, owing to the *NP complete* nature of the problem.

It is instructive to study facets of the maxcut polytope, since these provide good separation inequalities in a cutting plane approach. Numerous facets of the maxcut polytope have been studied, but the ones widely investigated in connection with a cutting plane approach are the following inequalities

$$\begin{array}{rcll} x_e & \geq & 0, & \forall e \in E, \\ x_e & \leq & 1, & \forall e \in E, \\ x(F) - x(C \backslash F) & \leq & |F| - 1 & \forall \text{ circuits } C \subseteq E \\ & & & \text{and all } F \subseteq C \text{ with } |F| \text{ odd}, \end{array} \tag{4}$$

where $x(S) = \sum_{\{i,j\} \in S} x_{ij}$ for any $S \subseteq E$. It is easy to see that these are valid inequalities for the maxcut polytope. The first and second set of inequalities define facets of $\mathrm{CUT}(G)$, if e is not contained in a triangle. The third set known as the *odd-cycle inequalities* defines facets for each $F \subseteq C$, with $|F|$ odd, if C has no chord. We will call a graph *chordal* if any cycle C of length ≥ 4 has a chord. The latter inequalities can be derived from the observation that every cycle and cut intersect in an even number of edges. Moreover, the integral vectors satisfying (4) are vertices of $\mathrm{CUT}(G)$. So we indeed have an integer programming formulation of the maximum cut problem, namely

$$\begin{array}{llcll} \max & & c^T x & & \\ \text{s.t.} & x(F) - x(C \backslash F) & \leq & |F| - 1 & \forall \text{ circuits } C \subseteq E \\ & & & & \text{and all } F \subseteq C \text{ with } |F| \text{ odd} \\ & x & \in & \{0,1\}^m. & \end{array} \tag{5}$$

Incidentally, the inequalities in (4) define a polytope known as the *odd cycle polytope*. Barahona and Mahjoub [4] show that we can drop the integrality restriction in (5), and solve the maxcut problem simply as an LP, if G does not have any subgraph contractible to K_5. This includes planar graphs, so we can indeed solve these maxcut instances in polynomial time.

Although there is an exponential number of linear constraints in (5), Barahona and Mahjoub [4] describe a polynomial time separation oracle for the inequalities (4). Thus it is possible to optimize a linear function over the odd cycle polytope in polynomial time. This exact algorithm, together with separation heuristics, has been used in cutting plane algorithms for the maxcut problem by Mitchell [35] and De Simone et al. [9, 10], among others. The initial relaxation used in these approaches is

$$\begin{array}{lccccc} \max & & & c^T x & & \\ \text{s.t.} & 0 & \leq & x & \leq & e, \end{array} \tag{6}$$

where $c, x \in \mathbb{R}^m$, where m is the number of edges in the graph.

3 Semidefinite formulations of the maxcut problem

In this section we consider semidefinite programming (SDP) formulations of the maxcut problem. Our polyhedral cut and price algorithm presented in §4 works with this SDP formulation.

The maxcut problem can be modelled as an integer program using cut vectors $d \in \{-1, 1\}^n$ with $d_i = 1$ if $i \in S$, and $d_i = -1$ for $i \in V \setminus S$. Consider the following problem

$$\begin{array}{lll} \max & \displaystyle\sum_{i,j=1}^{n} w_{ij} \frac{1 - d_i d_j}{4} & \\ \text{s.t.} & d_i^2 = 1, & i = 1, \ldots, n. \end{array} \tag{7}$$

A factor of $\frac{1}{2}$ accounts for the fact that each edge is considered twice. Moreover the expression $\frac{(1-d_i d_j)}{2}$ is 0 if $d_i = d_j$, i.e. if i and j are in the same set, and 1 if $d_i = -d_j$. Thus $\frac{(1-d_i d_j)}{2}$ yields the *incidence vector* of a cut associated with a cut vector d, evaluating to 1 if and only if edge $\{i, j\}$ is in the cut.

The Laplacian matrix of the graph G is $L := \text{Diag}(Ae) - A$, where A is the weighted adjacency matrix of the graph. The objective function in (7) can be rewritten as $\frac{1}{4} d^T L d$. Now, consider the symmetric matrix $X = dd^T$ of size n, whose (i, j)th entry is given by $X_{ij} = d_i d_j$. One can rewrite the maxcut problem as the following optimization problem over the matrix variable X

$$\begin{array}{llcl} \max & \frac{L}{4} \bullet X & & \\ \text{s.t.} & \mathrm{diag}(X) & = & e \\ & X & \succeq & 0 \\ & \mathrm{rank}(X) & = & 1. \end{array} \tag{8}$$

Dropping the rank one restriction on the matrix X gives the following SDP relaxation of the maxcut problem

$$\begin{array}{llcl} \max & \frac{L}{4} \bullet X & & \\ \text{s.t.} & \mathrm{diag}(X) & = & e \\ & X & \succeq & 0, \end{array} \tag{9}$$

and its dual

$$\begin{array}{llcl} \min & e^T y & & \\ \text{s.t.} & S & = & \mathrm{Diag}(y) - \frac{L}{4} \\ & S & \succeq & 0, \end{array} \tag{10}$$

which was originally derived in Laurent and Poljak [33] (see also Laurent [32], Helmberg [18] and Krishnan & Terlaky [29]). We will refer to the feasible region of (9) as the *elliptope*. It must be emphasized that the elliptope is no longer a polytope. Thus (9) is actually a non-polyhedral relaxation of the maxcut problem.

This semidefinite formulation of the maxcut problem satisfies strong duality, i.e., at optimality the objective values of (9) and (10) are the same. In a semidefinite cutting plane scheme for the maxcut problem, we would begin with (9) as our starting SDP relaxation.

Theorem 1 *The SDP relaxation (9) is tighter than the LP relaxation (6)*

Proof: We will show that any feasible solution in the SDP relaxation (9) gives a feasible solution in (6).

Consider any feasible solution X in (9). The constraints $\mathrm{diag}(X) = e$, and $X \succeq 0$ together imply that $-1 \leq X_{ij} \leq 1$.

We now transform the problem into $\{0, 1\}$ variables using the transformation $x_{ij} = \frac{1 - X_{ij}}{2}$. This gives $\frac{L}{4} \bullet X = \sum_{i=1}^{n} \sum_{i<j, \{i,j\} \in E} w_{ij} x_{ij}$. Also, we have $0 \leq x_{ij} \leq 1$, $\forall \{i, j\} \in E$. Such an x is feasible in (6). $\square$

Goemans and Williamson [13] developed a 0.878 approximation algorithm for the maxcut problem when all the edge weights are nonnegative. Their algorithm uses the solution X to the SDP relaxation (9), followed by an ingenious randomized rounding procedure to generate the incidence vector d of the cut. Since we employ the rounding scheme in our algorithm, we briefly discuss the procedure in §4.2. This 0.878 approximation ratio for the SDP formulation is a tremendous improvement over previous known LP formulations, where the ratio of the maxcut value to that of the LP relaxation over the odd cycle polytope could be $\frac{1}{2}$ for various sparse random graphs (Poljak and Tuza [42]). This ratio improves to $\frac{3}{4}$ if one considers dense random graphs, where it can be further improved by considering k-gonal inequalities; in fact one has a PTAS for dense graphs (Avis & Umemoto [2]). On the negative side, Håstad [17] showed that it is NP-hard to approximate the maxcut problem to within a factor of 0.9412.

If we include negative edge weights and still have the Laplacian matrix $L \succeq 0$, then Nesterov [36] showed that the GW rounding procedure gives an $\frac{2}{\pi}$ approximation algorithm for the maxcut problem.

The solution obtained by the randomized rounding procedure can be further improved using a Kernighan-Lin [24] (see also Papadimitriou and Steiglitz [37]) local search heuristic. Since we employ this heuristic too in our algorithm, we discuss it in §4.2.

One can further improve the relaxation (9) using chordless odd cycle inequalities. These are simply the odd cycle inequalities for the LP formulation in a ± 1 setting, and have the following form

$$\begin{array}{rcl} X(C\backslash F) - X(F) & \leq & |C| - 2 \\ & & \text{for each cycle } C, F \subset C, |F| \text{ odd}, \end{array} \tag{11}$$

where $X(S) = \sum_{\{i,j\} \in S} X_{ij}$ etc. These include among others the triangle inequalities. Since $\operatorname{diag}(X) = e$, and $X \succeq 0$ imply $-1 \leq X_{ij} \leq 1$, the feasible region of the SDP relaxation (9) intersected with the odd cycle inequalities (11) is contained within the odd cycle polytope. We should thus get tighter upper bounds on the maxcut value.

Interestingly, although the additional inequalities improve the SDP relaxation, they do not necessarily give rise to better approximation algorithms. On the negative side, Karloff [21] exhibited a set of graphs for which the optimal solution to (9) satisfies all the triangle inequalities as well, so after the Goemans-Williamson rounding procedure we are still left with a 0.878 approximation algorithm.

More recently, Anjos and Wolkowicz [1] presented a strengthened semidefinite relaxation of the maxcut problem. Their SDP relaxation is tighter than the relaxation (9) together with all the triangle inequalities. To arrive at their relaxation they add certain redundant constraints of the maxcut problem to (9), and consider the Lagrangian dual of this problem. On the negative side, this strengthened SDP relaxation is fairly expensive to solve. Also, see Lasserre [30, 31] for a hierarchy of increasingly tighter SDP relaxations for the maxcut problem.

4 An SDP based polyhedral cut and price method for the maxcut problem

This section deals with the major contribution of the paper which is a polyhedral cut and price algorithm, which is based on the SDP formulation of the maxcut problem presented in §3.

At the outset, we must mention that one could develop an SDP cutting plane approach for solving the maxcut problem. The approach sets up a series of primal SDP relaxations of the maxcut problem beginning with (9), which are progressively tightened by adding cutting planes, such as those presented in §3, to the primal SDP relaxation. Such an SDP cutting plane approach for the maxcut problem is discussed in Helmberg and Rendl [20].

Such an approach requires effective techniques for solving the SDPs that arise in the algorithm. The main tools for solving SDPs are interior point methods. Interior point methods, however, are fairly limited in the size of problems they can handle. Problems with more than 3000 constraints are considered quite hard (see Helmberg [18]).

Recently, several large scale approaches have been developed to try to overcome this drawback. For example, Helmberg [19] has incorporated the spectral bundle method in a cutting plane approach to solving the maxcut problem.

In this paper, we will solve each SDP in an interior point LP cutting plane framework. Such an approach is discussed in Krishnan and Mitchell [25, 26, 27, 28] and we present some details in §4.1. The overall approach is a polyhedral cut and price approach for the maxcut problem that mimics an SDP cutting plane approach. We note that in the past such cut and price LP approaches have been applied with a great degree of success in solving large scale integer programs (see Wolsey [44]).

We motivate our polyhedral cut and price approach as follows. The matrix variable X is equal to dd^T at optimality for (8), where d is a ± 1 vector representing a cut. Let d_i be a ± 1 vector representing a cut. We can write X as a convex combination of all such vectors, and so the maxcut problem (7) can also be written as follows

$$\begin{array}{lll} \max & \frac{L}{4} \bullet (\sum_{i \in \mathcal{I}} x_i d_i d_i^T) & \\ \text{s.t.} & \sum_{i \in \mathcal{I}} x_i d_{ij}^2 = 1, & j = 1, \ldots, n \\ & x_i \geq 0, & i \in \mathcal{I}, \end{array} \tag{12}$$

where $\mathcal{I}$ is the index set of all the ± 1 incidence cut vectors. Note that all the equality constraints in (12) are equivalent to the single constraint $\sum_i x_i = 1$ for this choice of index set $\mathcal{I}$. Moreover, $d_i^T d_i = n$, and there is an *exponential* number of such vectors. If we allow d_i instead to lie on the n dimensional sphere of radius $\sqrt{n}$, we obtain the following relaxation of the maxcut problem

$$\begin{array}{lll} \max & \frac{L}{4} \bullet (\sum_{i \in \mathcal{J}} x_i d_i d_i^T) & \\ \text{s.t.} & \sum_{i \in \mathcal{J}} x_i d_{ij}^2 = 1, & j = 1, \ldots, n \\ & x_i \geq 0, & \forall i \in \mathcal{J}, \end{array} \tag{13}$$

where $\mathcal{J}$ is the index set of all the vectors with Euclidean norm $\sqrt{n}$. In fact, (13) is exactly the SDP formulation (9) for the maxcut problem. Note that there are now *infinitely* many terms in the summation, so (13) is really a semi-infinite LP relaxation of the maxcut problem. Note that summing the constraints $\sum_{i \in \mathcal{J}} x_i d_{ij}^2 = 1$ implies $\sum_{i \in \mathcal{J}} x_i = 1$, so the feasible region is contained in the set of convex combinations of the vectors d_i.

We can solve (13) in a polyhedral column generation (pricing) scheme, with a small number of $d_i \in I\!R^n$ and generating others as necessary. This is reminiscent of the Dantzig-Wolfe decomposition approach applied to (13) and constitutes the pricing phase of the algorithm. The key to this pricing phase is the existence of a polynomial time separation oracle for the SDP relaxations. Of course, we eventually want to solve the problem over d_i that are ± 1 vectors. To do so, we introduce linear cutting planes in (13), that cut off some of the non ± 1 vectors not in the maxcut polytope; this is the cutting phase of the algorithm. We elaborate on these two phases below:

1. **Cutting phase:** The input to this phase is a feasible point X in (9). This is given to an oracle, which verifies whether this point is also within the maxcut polytope in a ± 1 setting. If this point is not in this polytope, the oracle returns linear cutting planes which are facets of the maxcut polytope. The cutting planes are added to the primal SDP relaxation, thereby strengthening it. This

is identical to the cutting phase of an SDP cutting plane approach, and the hope is that the added cutting planes enforce the rank one constraint (see the formulation (8) of the maxcut problem).

2. **Pricing phase:** The input to this phase is an SDP relaxation of the maxcut problem. We approximately solve the dual version of this SDP in a polyhedral interior point cutting plane scheme. In particular, one sets up LP relaxations of this dual SDP formulation, and approximately solves these relaxations using interior point methods. The solution is then given to a separation oracle for the SDP problem. The oracle verifies whether this point is also in the dual SDP feasible region. If this is not the case, the oracle returns linear cutting planes, which are supporting hyperplanes of the dual SDP feasible region. These cutting planes are added to the LP relaxation and the process is repeated. The reader familiar with linear programming decomposition will recognize the primal linear program as the reformulation obtained when Dantzig-Wolfe decomposition is applied to (9). Also, adding linear cutting planes in the dual LP relaxation amounts to pricing (column generation) in the primal LP setting. This is why we call this phase our pricing phase. We present some details in §4.1.

We now describe our entire algorithm in a nutshell. Each of these steps will be elaborated in more detail in §5.

Algorithm 1 (SDP cut-and-price algorithm for maxcut)

1. **Initialize**

2. **Approximately solve the dual maxcut SDP as an LP** *using the pricing phase discussed in §4.1. This gives an upper bound on the maxcut value. Construct the primal matrix X.*

3. **Generate good integer cut vectors** *by running the Goemans-Williamson and Kernighan-Lin heuristics on the primal matrix X. These heuristics are presented in §4.2. Add these integer vectors into the LP approximation, and solve this LP to optimality to find the new X.*

4. **Check for termination:** *If the difference between the upper bound and the value of the best cut is small, return the best cut as our solution to the maxcut problem. STOP.*

5. **Finding violated odd cycle inequalities:** *Feed the matrix X to a separation oracle that returns violated odd cycle inequalities. This oracle is described in Barahona and Mahjoub [4] and involves the solution of n shortest path problems on an auxiliary graph having twice the number of nodes and four times the number of edges. Bucket sort the resulting violated inequalities and add a subset of constraints to the primal LP approximation. This changes the dual slack matrix S.*

6. **Update the LP relaxation** *by choosing the most important constraints in the primal polyhedral approximation including the best integer cut vector, constraints based on the spectral factorization of X, and the initial box constraints. Return to Step 2.*

Assuming that we do not resort to branch and bound, at the termination of the cutting plane scheme we are approximately solving

$$\begin{array}{lrcl} \max & \frac{L}{4} \bullet X & & \\ \text{s.t.} & \operatorname{diag}(X) & = & e, \\ & \sum_{ij \in C \setminus F} X_{ij} - \sum_{ij \in F} X_{ij} & \leq & |C| - 2, \\ & \text{C} \ \text{ a cycle,} & F \subseteq C, & |F| \ \text{ odd,} \\ & X & \succeq & 0, \end{array} \tag{14}$$

which is still a relaxation of the maxcut problem.

We briefly motivate the steps in the algorithm: In Step 2 we are solving a relaxation of (14) in a polyhedral column generation scheme. We utilize this solution to get integer cut vectors in Step 3; we then add these integer vectors to the polyhedral approximation and resolve this LP. This ensures that some of the d_i in the LP are ± 1 vectors representing the best incidence cuts currently obtained by the algorithm. Moreover, Step 3 ensures that the feasible region of the LP approximation has a non-empty intersection with the maxcut polytope. In Step 5, we add violated odd cycle inequalities to the primal LP approximation; these odd cycle inequalities are cutting off some of the non ± 1 d_i vectors outside the maxcut polytope. Since we are employing an IPM to solve the LP approximations it is imperative that we keep the size of the LP approximations bounded. Step 6 realizes this objective by aggregating the useful information contained in the current polyhedral approximation.

We must mention that Gruber [16] has a similar approach, where he solves a semi-infinite formulation of the primal SDP relaxation (9). However, he does not motivate it in this way; his aim is rather to strengthen the LP cutting plane approach by adding cutting planes valid for the SDP approach. An advantage of his approach is that he deals exclusively with the primal problem. If the graph is chordal then it suffices to force every submatrix of X corresponding to a maximal clique to be positive semidefinite, so it is not necessary to consider the other entries in the primal matrix X explicitly. Grone et al. [14] showed that values for the missing entries can be chosen to make X positive semidefinite provided all these submatrices are positive semidefinite.

The approach is advantageous only if the underlying graph is chordal and sparse. If the graph is not chordal, then one needs to construct a chordal extension of it, by adding redundant edges of weight zero. This could lead to a fairly dense graph. Moreover, since one is confined to the edge set of the original graph in this approach, it is not clear how one could add hypermetric inequalities (see, eg, [11]) without enlarging the support of X.

4.1 An LP pricing approach for SDP

We describe the pricing phase (step 2 of Algorithm 1) in detail in this section.

Consider the following SDP

$$\begin{array}{lrcl} \max & C \bullet X & & \\ \text{s.t.} & A_i \bullet X & = & b_i, \quad i = 1, \ldots, m \\ & X & \succeq & 0, \end{array} \tag{15}$$

and its dual

$$\begin{array}{lrcl} \min & b^T y & & \\ \text{s.t.} & S & = & \sum_{i=1}^{m} y_i A_i - C, \\ & S & \succeq & 0, \end{array} \tag{16}$$

which are some of the relaxations of the maxcut problem encountered during the course of the algorithm.

Our method is based on the semi-infinite formulation of (16), namely

$$\begin{array}{lrcl} \min & b^T y & & \\ \text{s.t.} & S & = & \sum_{i=1}^{m} y_i A_i - C, \\ & d^T S d & \geq & 0, \quad \forall d \in B, \end{array} \tag{17}$$

where B is a compact set, typically $\{d : ||d||_2 \leq 1\}$ or $\{d : ||d||_\infty \leq 1\}$. We described a cutting plane approach to solving this relaxation in Krishnan and Mitchell [25, 26, 27, 28]. This approach sets up a relaxation to (16),

$$
\begin{array}{llll}
\min & b^T y & & \\
\text{subject to} & \sum_{j=1}^{m} y_j(d_i^T A_j d_i) & \geq & d_i^T C d_i \quad \text{for } i = 1, \dots, k,
\end{array}
\tag{18}
$$

with a corresponding constrained version of (15),

$$
\begin{array}{llll}
\max & C \bullet (\sum_{j=1}^{k} x_j d_j d_j^T) & & \\
\text{subject to} & A_i \bullet (\sum_{j=1}^{k} x_j d_j d_j^T) & = & b_i \quad i = 1, \dots, m \\
& x & \geq & 0.
\end{array}
\tag{19}
$$

We briefly present the pricing phase below:

Algorithm 2 (Pricing procedure)

1. **Initialize:** *Start with an initial relaxation of the form (18).*

2. **Solve the LP relaxation:** *Approximately solve (18) and (19) using a primal-dual interior point method to a prescribed tolerance for the solution (x, y).*

3. **Call the SDP separation oracle:** *Compute the slack matrix S at the approximate solution y to (18). If S is psd then we proceed to Step 4. Else, if S is indefinite, we add the following cutting plane*

$$\sum_{j=1}^{m} y_j(d^T A_j d) \quad \geq \quad (d^T C d),$$

where d is an eigenvector corresponding to the minimum eigenvalue of S to (18), update k, and return to Step 2.

4. **Tighten the tolerance:** *If the prescribed tolerance is still above the minimum threshold, tighten the tolerance, and proceed to Step 2. Else, the solution (X, y), where $X = \sum_{i=1}^{k} x_i d_i d_i^T$ is optimal for the current SDP relaxation. STOP.*

This cutting plane approach will be used as a subroutine to solve the SDP relaxations which arise while solving the maxcut problem.

An important point needs to be emphasized here: We use a primal-dual IPM to solve the LP relaxations approximately; the query points at which we call the oracle are more central iterates in the feasible region of (18), where this oracle returns better cuts. One could use the simplex method to solve the LP relaxations or even an interior point method solving these relaxations to optimality. This leads to the classical Kelley LP cutting plane scheme [23] for convex optimization (or the traditional Dantzig-Wolfe decomposition in the primal setting); here the query points are extreme points of the LP feasible region. It is known that the Kelley cutting plane method suffers from tailing effects, and we observed in Krishnan [25] (see also Elhedhli & Goffin [12]) that a simplex implementation performed very badly. Some convergence estimates for Kelley's method can be found in Lemarechal [34].

4.2 Heuristics for generating good incidence cut vectors

We briefly mention the Goemans-Williamson rounding procedure and the Kernighan-Lin local search heuristic in this section. These heuristics will be employed in generating good incidence cut vectors in step 3 of Algorithm 1.

Algorithm 3 (The Goemans-Williamson rounding procedure)

1. *We assume we have a feasible solution X in (9) to start with.*
2. *Compute Z such that $X = Z^T Z$. Let z_i, $i = 1, \ldots, n$ be the ith column of the matrix Z.*
3. *Generate a random vector r on the unit sphere in $\mathbb{R}^n$. Assign vertex i to S if $z_i^T r \geq 0$, and $V \backslash S$ otherwise. Repeat this procedure for all the vertices in the graph.*

In practice, one repeats Step 3 a number of times, and picks the best cut that was generated. We refer the reader to Goemans and Williamson [13] for more details.

Algorithm 4 (The Kernighan and Lin (KL) heuristic)

1. *We assume that we have a cut to start with; this is given by the bipartition S and $V \backslash S$ of the vertex set V. For each vertex i, $i = 1, \ldots, n$, compute the external and internal costs $E(i)$ and $I(i)$. For a vertex a in S this is the following (the other case is treated similarly):*

$$\begin{aligned} E(a) &= \sum_{i \in V \backslash S} w_{ai}, \\ I(a) &= \sum_{i \in S} w_{ai}, \end{aligned}$$

where w_{ij} is the weight of edge $\{i, j\}$.

2. *For each vertex i, compute $D^1(i) = E(i) - I(i)$, $i = 1, \ldots, n$. This can be interpreted as follows: Sending the vertex a from S to $V \backslash S$ will reduce the objective value of the cut by a factor of $D(a)$.*

3. *In the k iteration, choose a vertex b such that $D^k(b) = \min_i D^k(i)$ (not necessarily negative). Suppose that b goes from S to $V \backslash S$ (the other case can be handled similarly). Let $g(k) = D^k(b)$.*

4. *Mark b as* tabu, *implying that this vertex will not be considered in future interchanges. Update $D(v)$ for all vertices v adjacent to b as follows :*

$$\begin{array}{rcll} D^{k+1}(v) & = & D^k(v) + 2w_{bv}, & \forall v \in S, \\ D^{k+1}(v) & = & D^k(v) - 2w_{bv}, & \forall v \in V \backslash S. \end{array}$$

   Also, $D^{k+1}(v) = D^k(v)$, for all vertices v not adjacent to b.

5. *If $k < n$ return to step 3, else go to step 6.*

6. *When $k = n$, we are done considering all the vertices. We are back to the original cut, since we have swapped all the vertices, i.e. roles of S and $V \backslash S$ are reversed. Thus, we have $\sum_{i=1}^n g(i) = 0$.*

7. *Examine the cumulative function $G(k) = \sum_{i=1}^k g(i)$, $k = 1, \ldots, n$, and see where it is a minimum. If $G(k) \geq 0$, $k = 1, \ldots, n$ we stop. Else, choose k such that $G(k)$ is most negative (break ties arbitrarily), and perform the k first interchanges. This will give a cut, which is better than what we started with.*

One can repeat this procedure until it runs out of steam. In our implementation, we repeat the procedure 5 times in all. We refer the reader to Papadimitriou and Steiglitz [37] for more details on the K-L heuristic.

5 Details of the algorithm

In this section, we describe the steps of Algorithm 1 in more detail.

Our initial dual LP relaxation is

$$\begin{array}{llll} \min & \sum_{i=1}^{n} y_i & & \\ \text{s.t.} & y_i & \geq & \frac{L_{ii}}{4}, \quad i = 1, \ldots, n. \end{array} \tag{20}$$

The initial constraints in (20) define one-sided box constraints. They suffice for the maxcut problem since the objective function is the all ones vector, and an optimal solution is $(x^0, y^0, z^0) = (e, \mathrm{diag}(\frac{L}{4}), 0) \in I\!R^n \times I\!R^n \times I\!R^n$. This corresponds to initial matrices $X^0 = I$, and $S^0 = \frac{L}{4} - \mathrm{Diag}(\frac{L}{4})$. We note that for other problems we will in general need two sided box constraints to ensure that the initial relaxation has a bounded objective value.

The current dual maxcut SDP relaxation is not solved to optimality in step 2 of Algorithm 1; we mentioned some reasons in §4.1. We use a dynamically altered tolerance on the required accuracy of the solution (for specifics, see §6). However, if we do not solve the SDP accurately enough, then the objective value of the LP relaxation is not guaranteed to be an upper bound on the maxcut value. In fact we typically have the following inequalities on the objective values of the various relaxations.

$$\begin{array}{rcl} \text{Cheap LP approximation to SDP} & \leq & \text{Optimal Maxcut value} \quad \leq \\ \text{Good LP approximation to SDP} & \leq & \text{SDP objective value.} \end{array}$$

The accuracy to which the SDP is solved also affects the performance of the Goemans-Willamson rounding procedure. Using the same analysis as Goemans and Williamson [13], we can show that the value of the cut generated by the procedure is at least 0.878 times the value of the current LP approximation, if all the edge weights are nonnegative. However, we cannot always guarantee that the value of our LP approximation is an upper bound on the maxcut value. Thus, we do not have a performance guarantee out here. Nevertheless, we can say in a weak sense that the more accurately we solve the SDP as an LP in the cutting plane scheme, the more likely that a better cut is produced.

It should be noted that we add cutting planes in both the primal and dual spaces. We add cutting planes of the form (11) (based on the polyhedral structure of the maxcut polytope) in the primal space, while the cutting planes added in the dual are sometimes low dimensional faces of the dual SDP cone (in y space), and not always facets (see also Krishnan and Mitchell [28]); this amounts to column generation (pricing) in the primal space.

As regards the primal, we cannot guarantee that our primal LP feasible region always contains the entire maxcut polytope; in most cases it does not. The primal LP feasible region increases, when we solve the dual SDP as an LP (pricing), and decreases when we add cutting planes in the primal.

When we are adding cutting planes in the dual, we update the primal matrix $X = \sum_{i=1}^{m} x_i d_i d_i^T$. On the other hand, when we add cutting planes in the primal we are updating the dual slack matrix $S = \mathrm{Diag}(y) + \sum_{i=1}^{p} y_{n+i} A_i - \dfrac{L}{4}$, where A_i is the coefficient matrix associated with ith odd cycle inequality. In each iteration of the algorithm, we have an LP formulation of the dual maxcut SDP of the form

$$
\begin{array}{lrcl}
\min & \multicolumn{3}{c}{\displaystyle\sum_{i=1}^{n} y_i + \sum_{i=1}^{p}(|C_i|-2)y_{n+i}} \\
\text{s.t.} & \multicolumn{3}{c}{\begin{bmatrix} I & 0 & \dots & 0 \\ d_{n+1}.^2 & d_{n+1}^T A_1 d_{n+1} & \dots & d_{n+1}^T A_p d_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ d_m.^2 & d_m^T A_1 d_m & \dots & d_m^T A_p d_m \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ y_{n+1} \\ \vdots \\ y_{n+p} \end{bmatrix}} \\
& & \geq & \begin{bmatrix} \mathrm{diag}(\frac{L}{4}) \\ d_{n+1}^T \frac{L}{4} d_{n+1} \\ \vdots \\ d_m^T \frac{L}{4} d_m \end{bmatrix} \\
& y_{n+i} & \geq & 0, \quad i = 1, \dots, p,
\end{array}
\tag{21}
$$

and its dual

$$
\begin{array}{lrcll}
\max & \frac{L}{4} \bullet (\sum_{j=1}^{m} x_j d_j d_j^T) & & & \\
\text{s.t.} & \mathrm{diag}(\sum_{j=1}^{m} x_j d_j d_j^T) & = & e & \\
& A_i \bullet (\sum_{j=1}^{m} x_j d_j d_j^T) & \leq & (|C_i|-2), & i = 1, \dots, p \\
& x_j & \geq & 0, & j = 1, \dots, m.
\end{array}
\tag{22}
$$

An odd cycle inequality (11) is written as $A \bullet X \leq (|C|-2)$, where A is a symmetric matrix, and $|C|$ is the cardinality of the cycle. Also x_j is the primal variable corresponding to the jth dual constraint.

In step 2 of Algorithm 1, we improve the LP approximation to the SDP using the pricing strategy outlined in §4.1. The LPs are solved approximately using a primal-dual interior point method. Cutting planes are added to (21), which try and force the dual slack matrix $S = \mathrm{Diag}(y(1:n)) + \sum_{i=1}^{p} A_i y_{n+i} - \dfrac{L}{4}$ to be psd. Appropriate cuts can be found by looking for eigenvectors of the current dual slack matrix with negative eigenvalues. At the end of step 2, we have a solution (X, y), where X is feasible in the current primal SDP relaxation of the maxcut problem.

To ensure that the feasible region of the current LP approximation has a nonempty intersection with the maxcut polytope, we add good incidence cut vectors to the

LP approximation in step 3 of Algorithm 1. We run the Goemans and Williamson rounding procedure on X, and generate a number of good cut vectors. We then improve these cuts using the Kernighan and Lin (KL) heuristic. We then add the 5 best incidence vectors as cuts in (21). These heuristics were described in §4.2. We resolve the new LP for (X, y).

The value of the best cut found so far gives a lower bound on the optimal value of the maxcut problem. The value of any dual feasible vector $\hat{y}$ for which the corresponding slack matrix $\hat{S}$ is psd gives an upper bound on this optimal value. Such a vector can be constructed from the current solution y by increasing its first n components appropriately. To compute $\hat{y}$ we proceed as follows: Compute the dual slack matrix S at the point y. Let λ be the minimum eigenvalue of S. Then set

$$\begin{aligned} \hat{y}_i &= y_i - \lambda, \qquad i = 1, \ldots, n, \\ \hat{y}_i &= y_i, \qquad i = n+1, \ldots, n+p. \end{aligned}$$

The objective value at $\hat{y}$ gives us our current upper bound. The best such upper bound is stored, and when the upper bound and lower bound are close enough the algorithm is terminated in step 4 of Algorithm 1.

Cutting planes for (22) are found in step 5 of Algorithm 1. These odd cycle inequalities are violated by the current solution X. We use the exact Barahona-Mahjoub separation oracle which returns violated odd cycle inequalities (11), which are facets of the maxcut polytope. These odd cycle inequalities are bucket sorted by violation, and the most violated ones are added to (22).

In order to keep the size of the relaxations manageable, the constraint matrix is modified in step 6 of Algorithm 1. The rule is to drop those constraints whose primal variables x_j are zero or small. We sort these constraints based on their x values and retain the q (an upper bound is given below) best constraints ensuring that the best integer vector is among these constraints. The modified LP formulation (21) has the following constraints :

1. The box constraints, i.e. the first n constraints in the earlier (21).

2. The q best constraints from the earlier (21).

3. The eigenvectors v_i, $i = 1, \ldots, r$ corresponding to nonzero eigenvalues of X (r here is the rank of X). In a sense, the eigenvectors corresponding to the nonzero eigenvalues *aggregate* the important information contained in all the constraints added so far.

The number q is chosen as $q = \sqrt{2n} - r$. Note that, $\sqrt{2n}$ is an upper bound for the rank of at least one extreme point optimal solution for (14) with p odd cycle inequalities (see Pataki [38]).

The overall SDP cutting plane method for maxcut is best summarized in Figure 1. The figure illustrates how the primal LP feasible region evolves during the course of the algorithm. Since we are solving a *constrained* version of the primal SDP relaxation, the LP feasible region is always within the corresponding primal SDP cone. The polytope shaded is the maxcut polytope. Suppose we are in the kth iteration, and the LP feasible region is the polytope *abcdea*. Since we have solved the current dual SDP only approximately as an LP, the polytope *abcdea* does not contain the entire maxcut polytope. The Barahona separation oracle returns an odd cycle inequality, that is violated by the current LP solution x^{k+1}. The new LP feasible region is now the polytope *abcgha*. We then try and grow this LP feasible region, by adding cuts in the dual to try and force the new dual slack matrix S^{k+1} to be positive semidefinite. At the end of the process, let $abefgh$ be the new LP feasible region. Although, $abefgha$ does not contain the entire maxcut polytope either, it is a better approximation than the earlier polytope *abcdea*. Proceeding in this manner, we eventually hit the optimal vertex i.

6 Computational results

In this section we report our preliminary computational experience. All results were obtained on a *Sun Ultra 5.6*, *440 MHz* machine with *128 MB* of memory. We carried out our tests on a number of maxcut problems from *SDPLIB* [6], the *DIMACS Implementation Challenge* [39], and some randomly generated Ising spin glass problems, i.e. maxcut problems with ± 1 edge weights from Mitchell [35]. We compare our results with a traditional SDP cutting plane scheme that uses SDPT3 (Toh et al [43]) to solve the SDP relaxations.

We implemented our entire approach within the *MATLAB* environment. The LP solver was Zhang's *LIPSOL* [45]. The SDP separation oracle used in step 2 of Algorithm 1 employs MATLAB's Lanczos solver. Finally, we used the Barahona-Mahjoub separation oracle to generate valid cutting planes in the primal; our implementation was based on the *CPLEX* [8] network optimization solver to solve the shortest path problem as an LP. In practice, one uses heuristics (see Berry and Goldberg [5]) to generate the violated cuts, but we used the exact separation oracle in our experiments.

The computational results are summarized in Table 1. The columns in the table

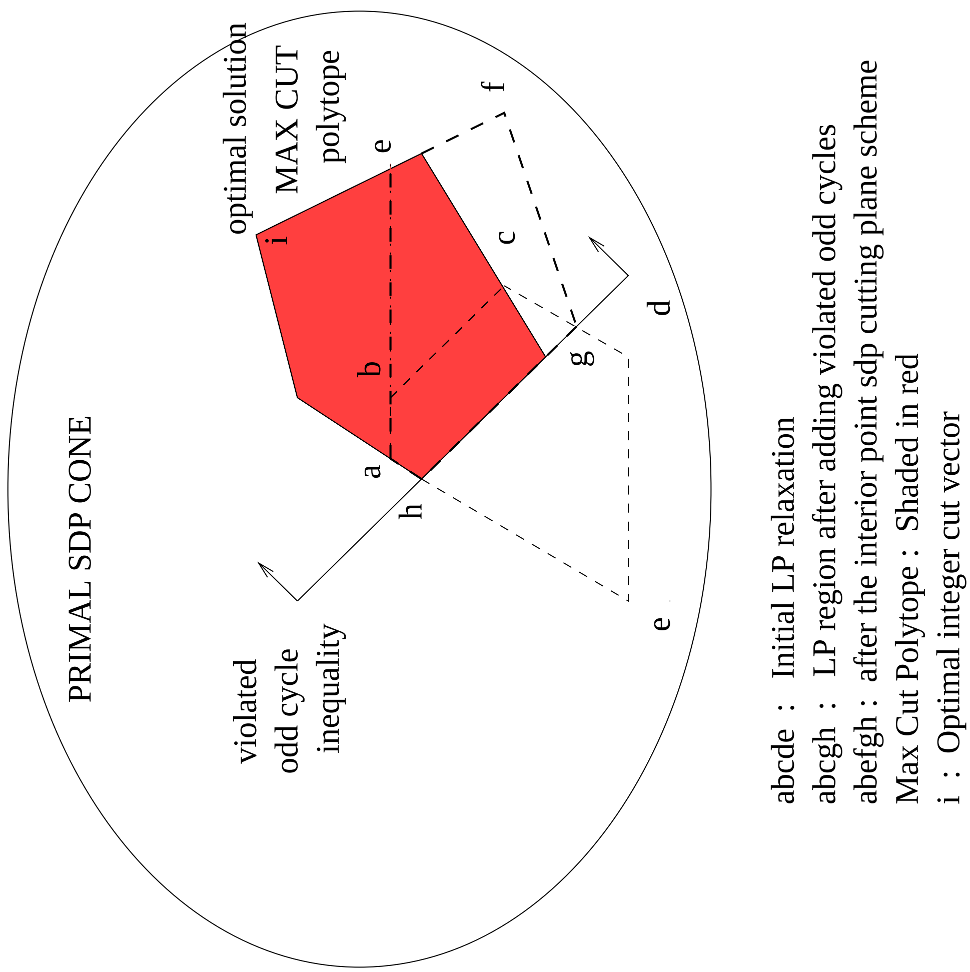

Figure 1: The polyhedral cut and price algorithm for the maxcut problem

represent

Name:	Problem name
n:	Number of nodes
m:	Number of edges
Maxcut:	Objective value of the best incidence vector found.
	If not optimal, the optimal value is noted.
SDP1:	Objective value over the elliptope (9)
SDP2:	Objective value over the elliptope and odd cycles (14)
UB:	The upper bound returned by the cutting plane scheme

Name	n	m	Maxcut(Opt)	SDP1	SDP2	UB
gpp100 [1]	100	269	210	221.69	211.27	212.56
gpp1241 [1]	124	149	137	141.91	137	137.45
gpp1242 [1]	124	318	256	269.64	257.66	258.60
gpp1243 [1]	124	620	446	467.68	458.35	458.90
gpp1244 [1]	124	1271	834	864.26	848.92	850.08
gpp2501 [1]	250	331	305	317.23	305	307.83
gpp2502 [1]	250	612	502	531.78	511.12	520.19
gpp5001 [1]	500	625	574	598.11	576.64 [4]	584.91
toruspm-8-50 [2]	512	1536	456	527.81	464.70 [4]	472.28
torusg3-8 [2]	512	1536	412.75 (416.84)	457.36	417.69 [4]	428.18
ising10 [3]	100	200	70	79.22	70	70.70
ising20_1 [3]	400	800	282	314.25	282	288.04
ising20_2 [3]	400	800	268	305.63	268	271.82
ising30 [3]	900	1800	630 (632)	709.00	632	

Table 1: Test Results on Maxcut

We ran 10 cutting plane iterations in all. In our LP subroutine for the SDP, we add 5 cutting planes in each iteration, our starting tolerance is 1, and this is lowered by a factor of 0.9 in each iteration. Initially, we solve our SDP relaxations very cheaply, i.e. perform only 5 LP cutting plane iterations, and we increase this number

[1]SDPLIB
[2]DIMACS
[3]Random Ising Spin glass problems (Mitchell)
[4]Best results obtained using an interior point approach

if we are unable to improve the best incidence cut vector. Also, we add the $\frac{n}{4}$ most violated odd cycle inequalities returned by the Barahona-Mahjoub approach in each iteration. Interestingly, we were able to obtain the optimal integer solution in most cases (except problems torusg3-8 and ising30). The former problem is not easy, since the maxcut solution is not integer, whereas the latter problem is currently as big as we can handle. For most of the problems we do not have a proof of optimality, since the SDP relaxation over the elliptope and odd cycles is not enough, and there is yet some gap involved. For the Ising spin glass problems optimizing over the intersection of the elliptope and the odd cycle polytope is sufficient to give the optimal solution.

To give an estimate of the times involved we performed the following experiment:

1. We ran our code against a standard SDP cutting plane scheme that employed $SDPT3$ [43] version 2.3 to solve the SDP relaxations.

2. We chose eight problems from SDPLIB.

3. We ran 10 cutting plane iterations in all.

4. In our approach, we solve the SDP relaxations very cheaply (5 LP cutting plane iterations), and in every 5th iteration we solve this SDP relaxation more accurately (25 iterations). In $SDPT3$ we solve the SDP relaxations to a moderate tolerance 10^{-2}, and in every 5th iteration we solve the SDP more accurately (10^{-6}).

5. We add the $\frac{n}{4}$ most violated odd cycle inequalities returned by the Barahona-Mahjoub approach in each iteration.

The results can be found in table 2. Table 2 shows how our cutting plane approach scales with the problem size, and the increase in runtime with problem size compares quite well with a traditional interior point SDP cutting plane approach. The interior point approach solved the problem gpp1241 to optimality in just 5 iterations. On the other hand on problem gpp5001, the interior point method was unable to complete the stipulated 10 iterations.

7 Conclusions

We have presented a polyhedral cut and price technique for approximating the maxcut problem; the polyhedral approach is based on a SDP formulation of the maxcut problem which is much tighter than traditional LP relaxations of the maxcut problem.

Problem Name	SDPT3			Our approach		
	UB	LB	Time	UB	LB	Time
gpp100	212.26	210	321	216.86	210	485
gpp1241	137	137	197	138.33	137	488
gpp1242	258.4	256	423	263.73	256	620
gpp1243	458.86	446	409	461.92	446	832
gpp1244	851.01	834	468	857.35	834	902
gpp2501	305	305	2327	309.95	304	1073
gpp2502	512.06	502	1739	521.06	502	1687
gpp5001	-	-	-	583.65	574	2893

Table 2: Comparing two SDP cutting plane schemes for maxcut

This linear approach allows one to potentially solve large scale problems. Here are some conclusions based on our preliminary computational results

1. It is worth reiterating that although our scheme behaves like an SDP approach for the maxcut problem, it is entirely a polyhedral LP approach. The SDP relaxations themselves are solved within a cutting plane scheme, so our scheme is really a cutting plane approach within another cutting plane approach. We refer to the inner cutting plane approach as our pricing phase.

2. The cutting plane approach is able to pick the optimal maxcut incidence vector quickly, but the certificate of optimality takes time.

3. We are currently working on a number of techniques to improve our upper bounding procedure using information from the eigenvector based on the most negative eigenvalue. Nevertheless, we believe that if we are able to handle this difficulty, then our approach can be effective on large scale maxcut problems.

4. The main computational task in our approach is the time taken by the SDP separation oracle. Another difficulty is that the oracle returns cutting planes that lead to dense linear programs. One way is to use a variant of the *matrix completion* idea in Gruber and Rendl [16], and examine the positive semidefiniteness of S_K for any clique $K \subseteq V$ (here S_K is a sub-matrix of S corresponding to only those entries in the clique). This should speed up the separation oracle, which will also return sparser constraints, since any cutting plane for S_k can be

lifted to $\mathcal{S}^n$ by assigning zero entries to all the components of the cutting plane not in K.

5. We are also trying to incorporate this approach in a branch and cut approach to solving the maxcut problem, i.e. we resort to branching when we are not able to improve our best cut values obtained so far, or our upper bounds.

6. Helmberg [19] shows that better SDP relaxations can be obtained by enlarging the number of cycles in the graph, i.e. adding edges of weights 0 between non-adjacent nodes. This gives rise to a complete graph, and the only odd cycles now are triangle inequalities, which can be inspected by complete enumeration. Again, this in sharp contrast to the LP case, where adding these redundant edges does not improve the relaxation. We hope to utilize this feature in our cutting plane approach, in the near future.

7. We hope to extend this approach to other combinatorial optimization problems such as min bisection, k way equipartition, and max stable set problems based on a Lovasz theta SDP formulation.

8 Acknowledgements

The authors would like to thank three anonymous referees whose comments greatly improved the presentation in the paper.

References

[1] M. F. Anjos and H. Wolkowicz. *Strengthened semidefinite relaxations via a second lifting for the max-cut problem.* Discrete Applied Mathematics, 119:79–106, 2002.

[2] D. Avis and J. Umemoto. *Stronger linear programming relaxations of max-cut.* Mathematical Programming, 97:451–469, 2003.

[3] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt. *An application of combinatorial optimization to statistical physics and circuit layout design.* Operations Research, 36:493–513, 1998.

[4] F. Barahona and A. R. Mahjoub. *On the cut polytope.* Mathematical Programming, 36:157–173, 1986.

[5] J. Berry and M. Goldberg. *Path optimization for graph partitioning problems.* Discrete Applied Mathematics, 90:27–50, 1999.

[6] B. Borchers. *SDPLIB 1.2, a library of semidefinite programming test problems.* Optimization Methods and Software, 11:683–690, 1999.

[7] K. C. Chang and D. Z. Du. *Efficient algorithms for layout assignment problems.* IEEE Transactions on Computer Aided Design, 6:67–78, 1987.

[8] CPLEX Optimization Inc. *CPLEX Linear Optimizer and Mixed Integer Optimizer.* Suite 279, 930 Tahoe Blvd. Bldg 802, Incline Village, NV 89541.

[9] C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, and G. Rinaldi. *Exact ground states of Ising spin glasses: New experimental results with a branch and cut algorithm.* Journal of Statistical Physics, 80:487–496, 1995.

[10] C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, and G. Rinaldi. *Exact ground states of two-dimensional $\pm J$ Ising spin glasses.* Journal of Statistical Physics, 84:1363–1371, 1996.

[11] M. M. Deza and M. Laurent. *Geometry of cuts and metrics.* Springer Verlag, Berlin Heidelberg, 1997.

[12] S. Elhedhli and J. L. Goffin, *The integration of an interior-point cutting plane method within a branch-and-price algorithm.* Mathematical Programming, 100:267–294, 2004.

[13] M. X. Goemans and D. P. Williamson. *Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming.* J. Assoc. Comput. Mach., 42:1115–1145, 1995.

[14] B. Grone, C.R. Johnson, E. Marques de Sa, and H. Wolkowicz. *Positive definite completions of partial Hermitian matrices.* Linear Algebra and its Applications, 58:109–124, 1984.

[15] M. Grötschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, Berlin, Germany, 1988.

[16] G. Gruber. *On semidefinite programming and applications in combinatorial optimization.* PhD thesis, University of Technology, Graz, Austria, April 2000.

[17] J. Håstad. *Some optimal inapproximability results.* Journal of the ACM, 48(4):798–859, 2001.

[18] C. Helmberg. *Semidefinite programming for combinatorial optimization.* Technical Report ZR-00-34, TU Berlin, Konrad-Zuse-Zentrum, Berlin, October 2000. Habilitationsschrift.

[19] C. Helmberg. *A cutting plane algorithm for large scale semidefinite relaxations.* In *The Sharpest Cut*, edited by M. Grötschel, Festschrift in honor of M. Padberg's 60th birthday, MPS-SIAM 2004, pp. 233-256.

[20] C. Helmberg and F. Rendl. *Solving quadratic (0,1)-problems by semidefinite programs and cutting planes.* Mathematical Programming, 82:291–315, 1998.

[21] H. Karloff. *How good is the Goemans-Williamson MAX CUT algorithm?* SIAM Journal on Computing, 29:336–350, 1999.

[22] R. M. Karp. *Reducibility among combinatorial problems.* In Complexity of Computer Computations, edited by R. E. Miller and J. W. Thatcher, Plenum Press, New York, 1972, pp. 85-103.

[23] J. E. Kelley. *The cutting plane method for solving convex programs.* Journal of SIAM, 8:703–712, 1960.

[24] B. W. Kernighan and S. Lin. *An efficient heuristic procedure for partitioning graphs.* Bell System Technical Journal, 49:291–307, 1970.

[25] K. Krishnan. *Linear programming approaches to semidefinite programming problems.* PhD thesis, Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, July 2002.

[26] K. Krishnan and J. E. Mitchell. *Semi-infinite linear programming approaches to semidefinite programming (SDP) problems.* Fields Institute Communications Series, Volume 37, *Novel approaches to hard discrete optimization problems*, edited by P. Pardalos and H. Wolkowicz, pages 121–140, 2003.

[27] K. Krishnan and J. E. Mitchell. *An unifying framework for several cutting plane methods for semidefinite programming.* AdvOL-Report No. 2004/2, Advanced Optimization Laboratory, McMaster University, August 2004 (to appear in Optimization Methods and Software, 2005).

[28] K. Krishnan and J. E. Mitchell. *Properties of a cutting plane method for semidefinite programming.* Technical report, Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, May 2003.

[29] K. Krishnan and T. Terlaky. *Interior point and semidefinite approaches in combinatorial optimization.* AdvOL-Report No. 2004/1, Advanced Optimization Laboratory, McMaster University, April 2004 (to appear in the GERAD 25th Anniversary Volume on *Graphs and Combinatorial Optimization*, edited by D. Avis, A. Hertz, and O. Marcotte, Kluwer Academic Publishers, 2005).

[30] J. B. Lasserre. *Global optimization with polynomials and the problem of moments.* SIAM Journal on Optimization, 11:796–817, 2001.

[31] J. B. Lasserre. *An explicit equivalent semidefinite program for nonlinear 0-1 programs.* SIAM Journal on Optimization, 12:756-769, 2002.

[32] M. Laurent. *A journey from some classical results to the Goemans Williamson approximation algorithm for maxcut.* Lecture given at CORE, Louvain La Neuve, September 1-12 2003. Available from http://homepages.cwl.nl/˜monique/

[33] M. Laurent and S. Poljak. *On a positive semidefinite relaxation of the cut polytope.* Linear Algebra and its Applications, 223:439–461, 1995.

[34] C. Lemarechal. *Non-differentiable Optimization* in Optimization, G.L. Nemhauser, A.H.G. Rinnooy Kan, and M.J. Todd, eds., North-Holland, New York, 529–572, 1989.

[35] J. E. Mitchell. *Computational experience with an interior point cutting plane algorithm.* SIAM Journal on Optimization, 10:1212–1227, 2000.

[36] Y. E. Nesterov. *Semidefinite relaxation and nonconvex quadratic optimization.* Optimization Methods and Software, 9:141–160, 1998.

[37] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications Inc., Mineola, New York, 1998.

[38] G. Pataki. *On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues.* Mathematics of Operations Research, 23:339–358, 1998.

[39] G. Pataki and S. H. Schmieta. *The DIMACS library of mixed semidefinite, quadratic, and linear programs.* Available at http://dimacs.rutgers.edu/Challenges/Seventh/Instances/

[40] R. Y. Pinter *Optimal layer assignment for interconnect.* Journal of VLSI Computational Systems, 1:123–137, 1984.

[41] S. Poljak and Z. Tuza. *Maximum cuts and large bipartite subgraphs.* In *Combinatorial Optimization*, volume 20 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS/DIMACS, pp. 181–244, 1995.

[42] S. Poljak and Z. Tuza. *The expected relative error of the polyhedral approximation of the maxcut problem.* Operations Research Letters, 16:191-198, 1994.

[43] K. C. Toh, M. J. Todd, and R. Tutuncu. *SDPT3– a Matlab software package for semidefinite programming.* Optimization Methods and Software, 11:545–581, 1999.

[44] L. Wolsey. *Integer Programming.* John Wiley & Sons, Inc., New York 1998.

[45] Y. Zhang. *User's guide to LIPSOL: Linear programming interior point solvers v0.4.* Optimization Methods and Software, 11:385–396, 1999.