Gap Inequalities for the Max-Cut Problem: A Cutting-Plane Algorithm

Laura Galli[1], Konstantinos Kaparis[2], and Adam N. Letchford[2]

[1] Warwick Business School, University of Warwick, United Kingdom
[2] Department of Management Science, Lancaster University, United Kingdom

Abstract. Laurent & Poljak introduced a class of valid inequalities for the *max-cut* problem, called *gap* inequalities, which include many other known inequalities as special cases. The gap inequalities have received little attention and are poorly understood. This paper presents the first ever computational results. In particular, we describe heuristic separation algorithms for gap inequalities and their special cases, and show that an LP-based cutting-plane algorithm based on these separation heuristics can yield very good upper bounds in practice.

1 Introduction

Given a graph $G = (V, E)$, and a vertex set $S \subset V$, the set of edges with exactly one end-vertex in S is called an *edge cutset* or *cut* and denoted by $\delta(S)$. In the *max-cut problem*, one is given the graph G, along with a vector of edge-weights, say $w \in \mathbb{Q}^{|E|}$. The task is to find a cut of maximum total weight, i.e., to find a set $S \subset V$ such that the quantity $\sum_{e\in\delta(S)} w_e$ is maximised.

The max-cut problem is a fundamental $\mathcal{NP}$-hard combinatorial optimisation problem, with a wide range of applications and connections to various branches of mathematics. For surveys, see Deza & Laurent [7], Laurent [16] and Poljak & Tuza [22].

A standard technique in combinatorial optimization is to associate a polytope (or more precisely a family of polytopes) with the problem under consideration (e.g., Cook *et al.* [5]). The polytope associated with the max-cut problem, called the *cut polytope*, has been studied intensively; see again Deza & Laurent [7].

Laurent & Poljak [18] introduced an interesting class of valid inequalities for the cut polytope, called *gap* inequalities. The gap inequalities are remarkably general, including several other important classes of inequalities (including some known to define facets) as special cases. Unfortunately, however, computing the right-hand side of a gap inequality, for a given left-hand side, is $\mathcal{NP}$-hard [18]. This suggests that it would be difficult to use gap inequalities as cutting planes (see also Avis [1]). Perhaps for this reason, the inequalities have received little attention from researchers.

In this paper, we show that, despite the above-mentioned difficulty, it is possible to use gap inequalities within an LP-based cutting-plane algorithm. There are, however, several issues that need to be overcome.

A.R. Mahjoub et al. (Eds.): ISCO 2012, LNCS 7422, pp. 178–188, 2012.

The structure of the paper is as follows. Section 2 is a literature survey. Section 3 presents various algorithms, including separation heuristics for the gap inequalities and their special cases, and a 'primal stabilisation' scheme. Section 4 presents some computational results. Section 5 is concerned with integrality ratios for small values of n. Finally, Section 6 contains some concluding remarks and future research directions.

2 Literature Review

Let n denote the number of vertices, and assume (without loss of generality) that the graph G is complete. For each edge $\{i, j\}$, let x_{ij} be a binary variable that takes the value 1 if and only if the edge $\{i, j\}$ lies in the cut. The max-cut problem then reduces to solving the following 0-1 Linear Program (0-1 LP):

$$\max \ \sum_{1 \le i < j \le n} w_{ij} x_{ij}$$
$$\text{s.t. } x_{ij} + x_{ik} + x_{jk} \le 2 \ (1 \le i < j < k \le n), \quad (1)$$
$$x_{ij} - x_{ik} - x_{jk} \le 0 \ (1 \le i < j \le n; k \ne i, j) \quad (2)$$
$$x_{ij} \in \{0,1\} \qquad (1 \le i < j \le n). \quad (3)$$

The cut polytope, denoted by CUT_n, is the convex hull in $\mathbb{R}^{\binom{n}{2}}$ of solutions to (1)-(3).

The gap inequalities for CUT_n take the following form:

$$\sum_{1 \le i < j \le n} b_i b_j x_{ij} \le \left(\sigma(b)^2 - \gamma(b)^2\right)/4 \qquad (\forall b \in \mathbb{Z}^n). \quad (4)$$

Here, $\sigma(b)$ denotes $\sum_{i \in V} b_i$, and

$$\gamma(b) := \min\left\{|z^T b| : z \in \{\pm 1\}^n\right\}$$

is the so-called *gap* of b. Note that the gap inequalities are infinite in number.

To see why the gap inequalities are valid, consider a max-cut instance in which $w_{ij} = b_i b_j$ for all $1 \le i < j \le n$. For any vertex set $S \subset V$, the weight of the cut $\delta(S)$ will be equal to the product of $\sum_{i \in S} b_i$ and $\sum_{i \in V \setminus S} b_i$. This quantity is maximised when both $\sum_{i \in S} b_i$ and $\sum_{i \in V \setminus S} b_i$ are as close to $\sigma(b)/2$ as possible. From the definition of $\gamma(b)$, this is achieved when $\sum_{i \in S} b_i = (\sigma(b) - \gamma(b))/2$ and $\sum_{i \in V \setminus S} b_i = (\sigma(b) + \gamma(b))/2$, or vice-versa, at which point the weight of the cut is equal to the right-hand side of the gap inequality. (This argument shows not only that the gap inequalities are valid, but also that every gap inequality defines a proper face of the cut polytope.)

As mentioned in the introduction, the gap inequalities include several other important classes of inequalities as special cases. They also dominate various other inequalities. Figure 1, taken from [10], gives a graphical representation of the situation. An arrow from one class to another means that the former is a generalization of, or dominates, the latter.

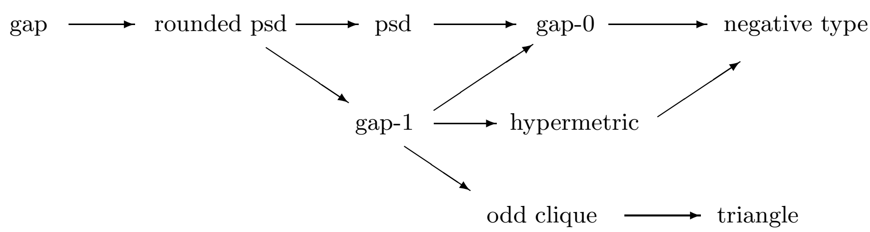

Fig. 1. Inequalities for the cut polytope

By 'gap-0' or 'gap-1' inequality, we simply mean a gap inequality with $\gamma(b)$ equal to 0 or 1, respectively. The other inequalities mentioned in the diagram are as follows:

– *Triangle* inequalities, which are nothing but the inequalities (1)-(2).
– *Negative-type* inequalities, obtained when $\sigma(b) = \gamma(b) = 0$ (Schoenberg [24]).
– *Hypermetric* inequalities, obtained when $\sigma(b) = \gamma(b) = 1$ (Deza [6] and Kelly [15]).
– *Odd clique* inequalities, obtained when $b \in \{0, \pm 1\}^n$ and $\sigma(b)$ is odd (Barahona & Mahjoub [3]).
– *Positive semidefinite* (psd) inequalities, which are the weakened version of gap inequalities obtained by replacing the right-hand side with $\sigma(b)^2/4$ (Laurent & Poljak [17]).
– *Rounded psd* inequalities, which are obtained by imposing that $\sigma(b)$ must be odd, and replacing the right-hand side with $\lfloor \sigma(b)^2/4 \rfloor$ (e.g., Avis & Umemoto [2], Giandomenico & Letchford [11], Letchford & Sørensen [19]).

So, the gap inequalities are extremely general.

We remark that the triangle and odd clique inequalities are facet-defining [3], and many rounded psd inequalities are too [7]. Moreover, it is shown in [17] that the psd inequalities define the feasible region of the well-known *Semidefinite Programming* (SDP) relaxation of the max-cut problem, which has been intensively studied (see, e.g., [12,21]). Therefore, the gap inequalities simultaneously generalise several classes of facet-defining inequalities and define a relaxation that dominates the SDP relaxation. So, one might expect them to be very useful as cutting planes.

Unfortunately, as pointed out in [17], computing $\gamma(b)$ is $\mathcal{NP}$-hard. Indeed, testing if $\gamma(b) = 0$ is equivalent to the *partition problem*, proven to be $\mathcal{NP}$-hard by Karp [14]. On the other hand, one can easily compute the gap in pseudo-polynomial time by dynamic programming. In our paper [10], we prove several complexity results about the gap inequalities and show that the associated separation problem can be solved in finite (though exponential) time. Nevertheless, to our knowledge, nobody had used gap inequalities computationally before the present paper.

Finally, we mention our other paper Galli *et al.* [9], in which we generalise the gap inequalities to the case of general non-convex mixed-integer quadratic programs.

3 Algorithms

This section concentrates on algorithmic aspects.

For a given class of inequalities, a *separation algorithm* is a procedure for detecting when inequalities in that class are violated. The following three subsections present various exact and heuristic separation algorithms for the gap inequalities and their special cases. In Subsection 3.4, we discuss some other ingredients of our cutting plane scheme.

3.1 A Separation Algorithm for Triangle Inequalities

The separation problem for the *triangle inequalities* can be solved easily, in $\mathcal{O}(n^3)$ time, by mere enumeration. We found, however, that a naive application of this idea causes a large number of triangle inequalities to be generated, which can significantly slow down the LP solver. Given that our cutting-plane scheme aims at finding good upper bounds in a reasonable amount of time, we decided to separate triangle inequalities heuristically. The following Algorithm 1 runs in $\mathcal{O}(n^2)$ time, and is guaranteed to generate no more than $\mathcal{O}(n^2)$ violated inequalities in any one call:

Algorithm 1. Heuristic separation algorithm for triangle inequalities

Input: A point $x^* \in \mathbb{R}^{\binom{n}{2}}$ to be separated.
Output: FAILURE or a violated triangle inequality.

for $1 \leq i < j \leq n$ **do**
  if $x^*_{ij} > (2+\epsilon)/3$ **then**
    Find $k : x^*_{ik} + x^*_{jk} = \max_{h=j+1\dots n}\{x^*_{ih} + x^*_{jh}\}$;
    if $x^*_{ij} + x^*_{ik} + x^*_{jk} \geq 2 + \epsilon$ **then**
      Return violated inequality $x_{ij} + x_{ik} + x_{jk} \leq 2$;
    end
  end
  if $x^*_{ij} > \epsilon$ **then**
    Find $k : x^*_{ik} + x^*_{jk} = \min_{h \in \{1\dots n\} \setminus \{i,j\}}\{x^*_{ih} + x^*_{jh}\}$;
    if $x^*_{ij} - x^*_{ik} - x^*_{jk} \geq \epsilon$ **then**
      Return violated inequality $x_{ij} - x_{ik} - x_{jk} \leq 0$;
    end
  end
end

Note that the role of ϵ in Algorithm 1 is that of defining the level of separation *precision*. In particular, the separation algorithm becomes exact if ϵ is set to 0.

3.2 Greedy Separation Heuristics

Helmberg & Rendl [13] present a useful greedy heuristic for the separation of *odd clique* inequalities. The basic idea consists in maintaining a collection of odd clique inequalities (some of which may be 'mere' triangle inequalities), from which new odd clique inequalities can be derived. For each inequality in the collection, the algorithm tries to extend the clique, by inserting two nodes, in the hope of obtaining a new violated odd clique inequality. Each time a new violated inequality is found, it is added to the collection.

One can extend the above greedy heuristic to handle the more general case of rounded psd inequalities. Now, one maintains a collection of rounded psd inequalities (some of which may be triangle and/or odd clique inequalities), from which new rounded psd inequalities can be derived. For each inequality in the collection, the algorithm attempts to increment or decrement two coefficients of the associated b vector, in the hope of obtaining a new violated rounded psd inequality. (To see why we have to modify two coefficients rather than one, recall that $\sigma(b)$ must be odd.)

The advantage of these greedy heuristics is twofold. First, the inequalities start out very sparse and get denser as the algorithm progresses, which eases the burden on the LP solver. Second, if one uses appropriate data structures, processing one inequality in the collection can be performed in $\mathcal{O}(n^2)$ time.

3.3 Separation Heuristics Based on Eigenvectors

Quite effective separation heuristics for rounded psd and gap inequalities can be derived by means of eigenvalue computations. The idea is to obtain an initial vector b, and then modify it, if necessary, in the hope of obtaining a violated inequality.

Our starting point is the following facts, due to Laurent & Poljak [17]. Let $x^* \in [0,1]^{\binom{n}{2}}$ be the point to be separated, and let $\mathcal{J}_n$ denote the convex body in $\mathbb{R}^{\binom{n}{2}}$ defined by the *psd* inequalities, mentioned in Section 2. If we construct a symmetric $n \times n$ matrix Y^* with $Y^*_{ii} = 1$ for all i and $Y^*_{ij} = 1 - 2x^*_{ij}$ for all $i \neq j$, the following fact holds:

$$x^* \in \mathcal{J}_n \Leftrightarrow Y^* \text{ is positive semidefinite.}$$

This implies that the separation problem for psd inequalities can be solved in polynomial time (to arbitrary fixed precision). To do this, it suffices to compute the minimum eigenvalue of Y^*, denoted by λ, along with the corresponding eigenvector b^*. If Y^* is not psd, then $\lambda < 0$, which implies that:

$$b^* Y^* b^* = b^* \cdot (\lambda b^*) = \lambda ||b^*||_2^2 < 0.$$

From the way in which Y^* was constructed, this implies that

$$\sum_{1 \le i < j \le n} b^*_i b^*_j x^*_{ij} > \sigma(b^*)^2/4,$$

and therefore we have found a violated psd inequality.

Clearly, given any violated or near-violated psd inequality, one can replace it with a rounded psd or gap inequality that is violated by at least as much. Note however that, in order to derive a rounded psd or gap inequality, the vector b must have integral components. On the other hand, most linear algebra software returns eigenvectors that have floating-point components. So, we have to apply some kind of scaling and rounding to the vector b^*. This leads to Algorithm 2.

Algorithm 2. Heuristic for finding useful b vectors

Input: A point $x^* \in \mathbb{R}^{\binom{n}{2}}$ to be separated.
Output: An initial b vector.

Let u be a prespecified upper bound on $||b||_1 = \sum_{i=1}^n |b_i|$;
Construct the matrix Y^*;
Let $b^* \in \mathbb{R}^n$ be an eigenvector of Y^* corresponding to the minimum eigenvalue;
Compute $u^* = ||b^*||_1$;
Scale b^* by multiplying it by u/u^*;
Round the components of b^* to integers: if the component is positive, round it down, otherwise round it up.

Once Algorithm 2 has been applied, we have an integral vector b whose norm $||b||_1$ is no larger than u. If $\sigma(b)$ is odd, we can generate a rounded psd inequality immediately. If $\sigma(b)$ is even, then one can check whether incrementing or decrementing one component of b leads to a violated rounded psd inequality.

If instead we wish to generate a gap inequality, then we must compute the gap $\gamma(b)$. As already stated, computing the gap is $\mathcal{NP}$-hard. Nevertheless, it can be done in pseudo-polynomial time, namely $\mathcal{O}(nu)$ time. Indeed, note that

$$\gamma(b) = ||b||_1 - 2\ \mathrm{SSP}, \tag{5}$$

where SSP is the solution to the following subset-sum problem:

$$\max \left\{ \sum_{i=1}^n |b_i| y_i : \sum_{i=1}^n |b_i| y_i \le \left\lfloor \frac{||b||_1}{2} \right\rfloor ,\ y \in \{0,1\}^n \right\}. \tag{6}$$

This subset-sum problem can be solved in $\mathcal{O}(n||b||_1)$ time by dynamic programming, which by construction is $\mathcal{O}(nu)$ time.

We highlight at this point that the scheme described above separates a *single* violated rounded psd or gap inequality per iteration, if it finds one. Our computational experience suggests that it is preferable to generate 'rounds' of rounded psd or gap inequalities per iteration instead of a single one. In fact, it is helpful to attempt to separate rounded psd or gap inequalities based on *all* n eigenvectors of the matrix Y^*. A possible explanation for the superiority of this framework lies in the *orthogonality* of the eigenvectors that are produced by most linear algebra solvers. This property seems likely to lead to 'diverse' rounded psd or gap inequalities (i.e., not close or parallel to each other), which in turn improves the performance of our cutting plane algorithm.

3.4 Some other Algorithmic Ingredients

Next, we briefly mention some additional algorithmic ideas that we have developed. The first of these has already proven to be extremely useful. We are still experimenting with the other three.

Primal Stabilisation. A pure LP-based cutting-plane algorithm based on gap inequalities suffers from the tailing-off phenomenon, which leads to very long running times for large instances. One way around it is to use what we call 'primal stabilisation', as follows:

1. Solve the SDP relaxation, yielding a solution x^*.
2. Add to the LP the 'box constraints' $x_e^* - \epsilon \leq x_e \leq x_e^* + \epsilon$ for all e, where $\epsilon > 0$ is a small parameter, to force the LP solution to stay near x^*.
3. Run the cutting-plane algorithm until the upper bound is close to the SDP bound, or until none of the box constraints are binding
4. Remove the box constraints, and continue with the cutting plane algorithm until some convergence criterion is reached.

This primal stabilisation scheme was inspired by the 'box-step' method of Marsten *et al.* [20], which is a classical technique for dual stabilisation in the context of Lagrangian relaxation and Dantzig-Wolfe decomposition. Of course, one can use our scheme only if an SDP solver is available.

Gap Strengthening. At the end of Algorithm 2 it is possible to check each component of b to see if adjusting it will lead to an increase in the violation if the gap inequality. Note that adjusting a given component b_i causes both the left-hand and right-hand sides of the gap inequality (4) to change. Using dynamic programming, we can compute the optimal adjustment of an individual component b_i in $\mathcal{O}(nu)$ time. So if we perform this procedure for $i = 1, \ldots, n$, this leads to an overall running time of $\mathcal{O}(n^2u)$.

Unfortunately, if we apply this strengthening procedure to an entire collection of gap inequalities, it may well happen that many of the strengthened inequalities turn out to be near-parallel. This is because the procedure attempts to make the violation as large as possible, so that, if in reality there exists only one gap inequality that is violated by the most, then the procedure will probably modify all of the original gap inequalities so that they all become similar to that one. As a result, they may perform poorly as a collection.

Another option, that we are planning to test, is to apply the procedure to just one of the gap inequalities in the collection.

Disjoint Triangle Packing. One major problem with triangle inequalities is that there are a huge number of them, and they can cause massive degeneracy in the LP. Thus, it seems reasonable to construct a *small* collection of triangle inequalities, which leads to a reasonably good upper bound, but does not slow down the LP solver too much. For this reason, we designed a procedure to construct an initial collection of triangle inequalities whose corresponding triangles are *edge-disjoint*. Note that such a collection can contain no more than $n(n-1)/6$ triangle inequalities. The choice of the initial set of disjoint triangles aims at the best improvement with respect to the trivial bound (i.e., the bound obtained if the LP contains only the trivial constraints $0 \le x_e \le 1$ for all e).

We are also experimenting with ways to update the triangle packing dynamically, after each cutting-plane iteration.

Sparsification of the *b* Vector. Finally, another problem with rounded psd and gap inequalities is that, if the b vector has many non-zero components, then the inequalities are very dense. LP solvers that are based on the simplex method are usually designed to exploit sparsity, and they don't cope very well with large numbers of dense constraints. It might be worth modifying Algorithm 2 in the following way. Once the eigenvector b^* is obtained, let $V^* \subset V$ be the set of nodes for which the component b_i^* differs significantly from zero. Then construct the principal submatrix of Y^* whose row and column set is V^*, and compute its minimum eigenvalue. The associated eigenvector can then be used to construct a rounded psd or gap inequality that has non-zero coefficients only for edges whose end-nodes are both in V^*.

We remark that a similar sparsification technique has been presented very recently by Qualizza *et al.* [23], but in the context of non-convex quadratically constrained quadratic programs.

4 Computational Results

In this section, we summarise the results obtained from the application of two cutting plane schemes on a subset of Max-Cut instances taken from the `Biq Mac` Library [1]. Among these instances, `g_05_n` are unweighted with edge probability 0.5 and $n = 60, 80, 100$. Instances `pm1d_80.0`, have edge probability 0.99, $w_{ij} = \{-1, 0, 1\}$ for $\{1 \le i \le j \le n\}$ and $n = 80$.

All our algorithms were implemented in ANSI C and tested on a PC Intel Xeon, with a 2.4 GHz processor and 12 GB of RAM, using ILOG CPLEX 12.1 as the LP solver. Computing times are expressed in seconds. Table 1 presents integrality gaps for different Max-Cut relaxations. Each row is the *average* of 10 instances of the given class. The first column corresponds to the integrality gap when one separates triangle inequalities *exactly* and solves the LP relaxation. The second column refers to the integrality gap for the SDP relaxation, which we computed using the CSDP package of Borchers [4]. The third column consist of two parts:

- *Mult.GAPs* is the integrality gap using gap inequalities only
- *Mult.GAPs+TIs* is the integrality gap using gap and triangle inequalities.

For the scheme *Mult.GAPs*, we used only the heuristic described in Subsection 3.3. This led to very poor convergence, and we had to abort each run while the

[1] `http://BiqMac.uni-klu.ac.at`

upper bound was still above the SDP bound. For the scheme *Mult.GAPs+TIs*, on the other hand, we used primal stabilisation, and also called the separation heuristic for gap inequalities only when the separation heuristic for triangle inequalities failed. This led to much better convergence and, as can be seen from the table, we were capable of obtaining significantly better bounds than those from the SDP relaxation, on all instances, within reasonable computing times.

Table 1. Average integrality gaps of various relaxations of the max-cut problem

	% IG TIs	% IG SDP	Mult.GAPs		Mult.GAPs+TIs	
			% IG	Time (scs)	% IG	Time (scs)
g_05_60	10.83	2.52	5.31	60	1.84	275
g_05_80	13.26	2.18	4.81	180	1.59	2685
g_05_100	15.28	2.23	5.01	600	1.93	15383
pmd1_80	102.1	22.3	35.55	180	15.12	3650

Due to limited space, we have not reported separate results for the odd clique and rounded psd inequalities. So far, we observed that (i) the bound obtained using odd clique inequalities is often not much better than the bound obtained using triangle inequalities alone and (ii) the bound obtained using rounded psd inequalities is usually almost as good as the one obtained using gap inequalities. It is not clear at this stage if these phenomena are due to the nature of the inequalities themselves, or due to the fact that we are using heuristics for separation, rather than exact algorithms.

5 Integrality Ratios Using Gap Inequalities

For a given n, let F be any family of valid inequalities for the cut polytope CUT_n. For any given edge-weight vector $w \in \mathbb{Q}^{\binom{n}{2}}$, one can compute an upper bound for the max-cut problem by optimising over the convex set defined by the inequalities in F. We define the *integrality ratio* of F as the supremum, over all possible non-negative and non-zero vectors w, of the ratio between the upper bound and the weight of the optimal cut. (We exclude vectors with negative entries, along with the zero vector, to avoid the possibility that the optimal cut has weight zero.)

In Galli & Letchford [8], exact integrality ratios were computed for $3 \leq n \leq 7$ for various families of inequalities, including triangle, odd clique, and rounded psd inequalities. Also a lower bound on the integrality ratio for $n = 8$ was presented. Using the procedures described in [8], together with the exact separation algorithm for gap inequalities described in Galli *et al.* [10], we were able to compute integrality ratios for gap inequalities as well. Table 2 displays the results in [8], along with our new results (highlighted in dark-gray). The column for $n = 8$ is marked with an asterisk to highlight the fact that the numbers in that column are lower bounds.

Table 2. Integrality ratios of various relaxations of the max-cut problem, for small values of n

n	3	4	5	6	7	8*
trivial	3/2	3/2	5/3	5/3	7/4	7/4
	1.5	1.5	1.667	1.667	1.75	1.75
triangle	1	1	10/9	10/9	7/6	7/6
	1	1	1.111	1.111	1.167	1.167
odd clique	1	1	1	25/24	21/20	21/20
	1	1	1	1.042	1.05	1.05
rounded psd	1	1	1	1	31/30	31/30
	1	1	1	1	1.033	1.033
gap	1	1	1	1	31/30	31/30
	1	1	1	1	1.033	1.033

Perhaps surprisingly, for the values of n considered, the gap inequalities provided the same ratios as the rounded psd inequalities (highlighted in light-gray). Indeed, we have not yet found a gap inequality that is not implied by one or more rounded psd inequalities. On the other hand, it is proved in [10] that such gap inequalities should exist, unless $\mathcal{NP}$ equals co-$\mathcal{NP}$.

6 Conclusions

This paper represents a first algorithmic and computational study on a cutting-plane scheme for the max-cut problem based on gap inequalities. To our knowledge it also constitutes the first cutting plane algorithm which is based solely on linear programming techniques and yet improves upon the semidefinite bound. This was achieved through primal stabilisation, combined with the separation of triangle and gap inequalities. Future research will be devoted to further testing the effectiveness of odd-clique and rounded-psd inequalities. Also, more effort will be put into exploring the algorithmic enhancements mentioned in Subsection 3.4.

References

1. Avis, D.: On the Complexity of Testing Hypermetric, Negative Type, k-Gonal and Gap Inequalities. In: Akiyama, J., Kano, M. (eds.) JCDCG 2002. LNCS, vol. 2866, pp. 51–59. Springer, Heidelberg (2003)
2. Avis, D., Umemoto, J.: Stronger linear programming relaxations of max-cut. Math. Program. 97, 451–469 (2003)
3. Barahona, F., Mahjoub, A.R.: On the cut polytope. Math. Program. 36, 157–173 (1986)

4. Borchers, B.: CSDP, a C library for semidefinite programming. Optim. Meth. & Software 11, 613–623 (1999)
5. Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A.: Combinatorial Optimization. Wiley, New York (1998)
6. Deza, M.: On the Hamming geometry of unitary cubes. Soviet Physics Doklady 5, 940–943 (1961)
7. Deza, M.M., Laurent, M.: Geometry of Cuts and Metrics. Springer, Berlin (1997)
8. Galli, L., Letchford, A.N.: Small bipartite subgraph polytopes. Oper. Res. Lett. 38(5), 337–340 (2010)
9. Galli, L., Kaparis, K., Letchford, A.N.: Gap inequalities for non-convex mixed-integer quadratic programs. Oper. Res. Lett. 39(5), 297–300 (2011)
10. Galli, L., Kaparis, K., Letchford, A.N.: Complexity results for the gap inequalities for the max-cut problem. To appear in Oper. Res. Lett (2011)
11. Giandomenico, M., Letchford, A.N.: Exploring the relationship between max-cut and stable set relaxations. Math. Program. 106, 159–175 (2006)
12. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. Ass. Comp. Mach. 42, 1115–1145 (1995)
13. Helmberg, C., Rendl, F.: Solving quadratic (0,1)-programs by semidefinite programs and cutting planes. Math. Program. 82, 291–315 (1998)
14. Karp, R.: Reducibility among combinatorial problems. In: Miller, R., Thatcher, J. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press, New York (1972)
15. Kelly, J.B.: Hypermetric spaces. In: The Geometry of Metric and Linear Spaces. Lecture Notes in Mathematics, vol. 490, pp. 17–31. Springer, Berlin (1974)
16. Laurent, M.: Max-cut problem. In: Dell'Amico, M., Maffioli, F., Martello, S. (eds.) Annotated Bibliographies in Combinatorial Optimization, pp. 241–259. Wiley, Chichester (1997)
17. Laurent, M., Poljak, S.: On a positive semidefinite relaxation of the cut polytope. Lin. Alg. Appl. 223/224, 439–461 (1995)
18. Laurent, M., Poljak, S.: Gap inequalities for the cut polytope. SIAM Journal on Matrix Analysis 17, 530–547 (1996)
19. Letchford, A.N., Sørensen, M.M.: Binary Positive Semidefinite Matrices and Associated Integer Polytopes. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) IPCO 2008. LNCS, vol. 5035, pp. 125–139. Springer, Heidelberg (2008)
20. Marsten, R.E., Hogan, W.W., Blankenship, J.W.: The `BOXSTEP` method for large-scale optimization. Oper. Res. 23, 389–405 (1975)
21. Poljak, S., Rendl, F.: Nonpolyhedral relaxations of graph bisection problems. SIAM J. on Opt. 5, 467–487 (1995)
22. Poljak, S., Tuza, Z.: Maximum cuts and large bipartite subgraphs. In: Cook, W., Lovász, L., Seymour, P. (eds.) Combinatorial Optimization. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 20, pp. 181–244. American Mathematical Society (1995)
23. Qualizza, A., Belotti, P., Margot, F.: Linear programming relaxations of quadratically constrained quadratic programs. In: Lee, J., Leyffer, S. (eds.) Mixed Integer Nonlinear Programming. IMA Volumes in Mathematics and its Applications, vol. 154, pp. 407–426. Springer, Berlin (2012)
24. Schoenberg, I.J.: Metric spaces and positive definite functions. Trans. Amer. Math. Soc. 44, 522–536 (1938)