A Branch and Bound Algorithm for Max-Cut based on combining semidefinite and polyhedral relaxations*

Franz Rendl† Giovanni Rinaldi‡ Angelika Wiegele§

November 9, 2006

Keywords: branch and bound algorithms, graph and network algorithms, integer programming, semidefinite programs.

Abstract

In this paper we present a method for finding exact solutions of the Max-Cut problem $\max x^T Lx$ such that $x \in \{\pm 1\}^n$. We use a semidefinite relaxation combined with triangle inequalities, which we solve with the bundle method. This approach is due to Fischer, Gruber, Rendl, and Sotirov [12] and uses Lagrangian duality to get upper bounds with reasonable computational effort. The expensive part of our bounding procedure is solving the basic semidefinite programming relaxation of the Max-Cut problem.

We review other solution approaches and compare the numerical results with our method. We also extend our experiments to unconstrained quadratic 0-1 problems and to instances of the graph bisection problem.

The experiments show, that our method nearly always outperforms all other approaches. Our algorithm, which is publicly accessible through the Internet, can solve virtually any instance with about 100 variables in a routine way.

*Supported in part by the EU project Algorithmic Discrete Optimization (ADONET), MRTN-CT-2003-504438.

†Alpen-Adria-Universität Klagenfurt, Institut für Mathematik, Universitätsstr. 65-67, 9020 Klagenfurt, Austria, franz.rendl@uni-klu.ac.at

‡Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" – CNR, Viale Manzoni, 30, 00185 Roma, Italy, rinaldi@iasi.cnr.it

§Alpen-Adria-Universität Klagenfurt, Institut für Mathematik, Universitätsstr. 65-67, 9020 Klagenfurt, Austria, angelika.wiegele@uni-klu.ac.at

1 Introduction

The Max-Cut problem is one of the fundamental NP-hard combinatorial optimization problems. It corresponds to unconstrained quadratic optimization in binary variables. We will present an exact method for this problem, which allows us to solve instances of modest size (about 100 binary variables) in a routine manner.

Since the late 1980's a systematic investigation based on polyhedral combinatorics was carried out to get exact solutions of the Max-Cut problem (see, e.g., [2, 3, 9, 11, 23, 1]). This approach is quite successful on sparse instances (e.g., in [9] the solution of toroidal grid graphs of sizes up to 22,500 nodes is reported), but it becomes no more usable for dense instances with more than, say, 50 nodes.

A major theoretical break-through occured in the early 1990's, when Goemans and Williamson [16] showed that a semidefinite programming (SDP) relaxation of Max-Cut has an error of no more than about 14%, independent of the density of the underlying problem, provided the edge weights in the problem are all nonnegative. This raised the hope that the use of this relaxation might open the way to deal also with dense instances. Unfortunately, this SDP bound is still too weak, see Poljak and Rendl [28]. Closing an initial gap of more than 10% by Branch and Bound is very likely to produce a huge number of subproblems to be investigated, leading to excessive computation times.

In this paper we take up the approach from Helmberg and Rendl [18] of using this SDP bound tightened by the inclusion of triangle inequalities in a Branch and Bound framework. The major improvement as compared to Helmberg and Rendl [18] consists in the way we compute the resulting relaxation. We use the approach from [12], which combines an interior-point method to compute the basic SDP relaxation with the bundle method to handle the triangle inequalities, and which we tuned for the Branch and Bound setting. A similar approach, but based on a pure polyhedral relaxation, was used quite successfully in [13] to compute the bound based on the triangle inequalities very effectively. We report computational results with this approach on a wide variety of instances and compare with virtually all existing methods. With the exception of very sparse graphs, our approach is a substantial improvement over all existing methods to solve the Max-Cut problem to optimality.

The paper is organized as follows. After a quick introduction to the problem (Section 2), we describe the SDP bound enhanced with triangle inequalities in Section 3. In Section 4 we briefly touch the other features of our Branch and Bound approach. We test our approach on a variety of data sets. Some characteristics of these data along with their origin are given in Section 5. In Section 6 we compare our approach with existing exact methods. Finally we discuss some extensions of our approach to the graph equipartition problem.

Notation: We use standard notation from graph theory. The vector of all ones (of appropriate dimension) is denoted by e, $\mathcal{A}$ is a linear operator mapping symmetric matrices to vectors in $\mathbb{R}^m$, and $\mathcal{A}^T$ is its adjoint operator. For a vector v of size n we denote by $\mathrm{Diag}(v)$ the matrix D of order n with $D_{ii} = v_i$ and with all the off-diagonal elements equal to zero. For a matrix D or order n, $\mathrm{diag}(D)$ denotes the n-dimensional vector v with $v_i = D_{ii}$. Finally, tr D denotes the trace of the square matrix D, i.e., the sum of its diagonal elements.

2 The Max-Cut problem

The Max-Cut problem is one of the basic NP-hard problems and has attracted scientific interest from the combinatorial optimization community, and also from people interested in nonlinear optimization. There are two essentially equivalent formulations of the problem.

Max-Cut in a graph: Given an undirected graph $G = (V, E)$ on $|V| = n$ vertices with edge weights w_e for $e \in E$, every bipartition (S, T) of V (where S or T can be empty) defines a cut $(S : T) = \{ij \in E : i \in S, j \in T\}$. The problem is to find a bipartition (S, T) such that the weight of the corresponding cut

$$w(S, T) := \sum_{e \in (S:T)} w_e$$

is maximized. It will be convenient to use matrix notation and introduce the weighted adjacency matrix $A = (a_{ij})$ with $a_{ij} = a_{ji} = w_e$ for edge $e = [ij] \in E$ and $a_{ij} = 0$ if $[ij] \notin E$. Given A we also introduce the matrix L defined by $L = \mathrm{Diag}(Ae) - A$, often called the Laplacian, associated to A.

If we represent bipartitions (S, T) by vectors $x \in \{\pm 1\}^n$ with $x_i = 1$ exactly if $i \in S$, then it is easy to show that $w(S, T) = \frac{1}{4} x^T L x$. Hence finding a cut in a graph with maximum weight is equivalent to solving the following quadratic optimization problem.

$$\text{(MC)} \quad z_{MC} = \max\{x^T L x : \ x \in \{\pm 1\}^n\}.$$

Quadratic 0-1 minimization: Given a matrix Q of order n and a vector c, let $q(y) := y^T Q y + c^T y$. We consider the following problem.

$$\text{(QP)} \quad \min\{q(y) : \ y \in \{0, 1\}^n\}.$$

It is not difficult to show that solving (QP) is equivalent to solving (MC) (see for instance [3]). We consider both models, as both are dealt with in the literature.

3 Semidefinite relaxations of (MC)

The following semidefinite relaxation of (MC) uses $x^T L x = \mathrm{tr} L(xx^T)$ and introduces a new matrix variable X taking the role of xx^T.

$$z_{SDP} = \max\{\mathrm{tr}\ LX : \mathrm{diag}(X) = e, \ X \succeq 0\}. \tag{1}$$

Its dual form

$$\min\{e^T u : \mathrm{Diag}(u) - L \succeq 0\} \tag{2}$$

was introduced by Delorme and Poljak [10] as the (equivalent) eigenvalue optimization problem

$$\min\{n\lambda_{\max}(L - \mathrm{Diag}(u)) : u \in \mathbb{R}^n, u^T e = 0\}. \tag{3}$$

The primal version (1) can be found in [28]. In [16] it is shown that this relaxation has an error of no more than 13.82%, i.e.,

$$\frac{z_{SDP}}{z_{MC}} \leq 1.1382,$$

provided there are non-negative weights on the edges ($w_e \geq 0$). This relaxation can be further tightened by including the following triangle inequalities (that define the *semimetric polytope*, the basic polyhedral relaxation of Max-Cut).

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_{ij} \\ x_{ik} \\ x_{jk} \end{pmatrix} \leq \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad 1 \leq i < j < k \leq n.$$

We abbreviate all $4\binom{n}{3}$ of these constraints as $\mathcal{A}(X) \leq e$. Hence we get

$$z_{SDPMET} = \max\{\mathrm{tr}\ LX : \mathrm{diag}(X) = e, \ \mathcal{A}(X) \leq e, \ X \succeq 0\}. \tag{4}$$

Helmberg and Rendl [18] apply this semidefinite relaxation (solved by an interior point code) in a Branch and Bound scheme. Later on, Helmberg [17] improved this algorithm by fixing variables. The experiments in [18] clearly indicate that an efficient computation of this relaxation is crucial for further computational improvements.

Instead of solving this relaxation with a limited number of inequality constraints by interior point methods, as done in [18], we use the bundle approach, suggested in [12], which we modify to gain computational efficiency in the Branch and Bound process.

The set $\mathcal{E} := \{X : \mathrm{diag}(X) = e, X \succeq 0\}$ defines the feasible region of (1). Therefore (4) can compactly be written as

$$z_{SDPMET} = \max\{\langle L, X\rangle : X \in \mathcal{E}, \mathcal{A}(X) \leq e\}. \tag{5}$$

We now briefly recall the approach from [12] to approximate z_{SDPMET} (from above). Let us introduce the Lagrangian with respect to $\mathcal{A}(X) \leq e$

$$\mathcal{L}(X, \gamma) := \langle L, X\rangle + \gamma^T(e - \mathcal{A}(X)) \tag{6}$$

and the associated dual function

$$f(\gamma) := \max_{X\in\mathcal{E}} \mathcal{L}(X, \gamma) = e^T\gamma + \max_{X\in\mathcal{E}} \langle L - \mathcal{A}^T(\gamma), X\rangle. \tag{7}$$

We get for any $\hat{\gamma} \geq 0$ that

$$z_{SDPMET} = \max_{X\in\mathcal{E}} \min_{\gamma\geq 0} L(X, \gamma) = \min_{\gamma\geq 0} f(\gamma) \leq f(\hat{\gamma}).$$

The problem now consists in finding a 'good' approximation $\hat{\gamma}$ to the correct minimizer of f.

The function f is well-known to be convex but non-smooth. Evaluating f for some $\gamma \geq 0$ amounts to solving a problem of type (1), which can be done easily for problem sizes of our interest. We use a primal-dual interior-point method to solve it, which also provides an optimality certificate X_γ, u_γ (optimal solutions to (1) and (2)). The primal matrix X_γ will turn out to be useful in our algorithmic setup. We have, in particular that

$$f(\gamma) = \mathcal{L}(X_\gamma, \gamma).$$

Moreover, a subgradient of f at γ is given by $e - \mathcal{A}(X_\gamma)$.

Dualizing all triangle constraints would result in a dual problem of dimension roughly $\frac{2}{3}n^3$. We prefer a more economical approach where inequalities are included only if they are likely to be active at the optimum.

Let I be a subset of the triangle inequalities, hence $\mathcal{A}_I(X) \leq e_I$. We also write γ_I for the variables dual to the inequalities in I. Setting the dual variables not in I to zero, it is clear that for any I and any $\gamma_I \geq 0$, we get an upper bound on z_{SDPMET}. Approximating the value z_{SDPMET} therefore breaks down into the following two independent tasks:

1. Identify a subset I of triangle inequalities.
2. For a given set I of inequalities, determine an approximate minimizer $\gamma_I \geq 0$ of f.

The second step can be carried out with any of the subgradient methods for convex nonsmooth functions. For computational efficiency we use the bundle method with a limit on the number of function evaluations.

Carrying out the first step is less obvious. We are interested in constraints which are active at the optimum, but this information is in general not available. Therefore we use the optimizer X_{γ_I}, corresponding to an approximate minimizer γ_I of f, and add to the current set I of constraints the

t triangle inequalities most violated by X_{γ_I}. (Here t is a parameter which is dynamically chosen.) Thus we can identify promising new inequalities to be added to I.

On the other hand, we remove any constraint from I where the dual multiplier is close to zero, as this is an indication that the constraint is unlikely to be binding. We iterate this process of selecting and updating a set of triangle inequalities, and then solving the respective relaxation, as long as the decrease of the upper bound is sufficiently large.

4 Branching rules and heuristics

4.1 Branching strategies

We subdivide the set of feasible solutions by simply separating, or merging two vertices i, j. This results again in an instance of (MC), see [27]. There are several natural choices for such a pair i, j for branching.

• **Easy first:**
A first idea is to branch on pairs i, j where the decision seems to be obvious. We choose i and j such that their rows are 'closest' to a ± 1 vector, i.e., they minimize $\sum_{k=1}^{n}(1 - |x_{ik}|)^2$. We may assume, that for these two very well articulated nodes the value $|x_{ij}|$ is also very large. Setting x_{ij} opposite to its current sign should lead to a sharp drop of the optimal solution in the corresponding subtree. Hoping that the bound also drops as fast, we will, presumably, be able to cut off this subtree quickly. This rule has been used also in [18] and called R2.

• **Difficult first:**
Another possibility for branching is to fix the hard decisions first. We branch on the pair i, j which minimizes $|x_{ij}|$. This means, we fix the most difficult decisions and hope that the quality of the bound gets better fast and that the subproblems become easier. Following [18] we call this rule R3.

Depending on the class of problems, either rule R2 or R3 was more efficient than the other. We also experimented with the so-called *strong branching*, as this strategy is quite successful for linear programming based relaxations. Unfortunately, sensitivity information, necessary for selecting the branching pair, is much harder to get in the case of semidefinite relaxations, hence there is no computational trade off. Consequently, we did not pursue this strategy any further.

4.2 Generating feasible solutions

Generating feasible solutions is done iteratively in basically three steps:

1. Apply the Goemans-Williamson hyperplane rounding technique [16] to the primal matrix X obtained from solving the SDP during the bundle iterations. This gives a cut vector $\bar{x}$.
2. Cut $\bar{x}$ is locally improved by checking all possible moves of a single vertex to the opposite partition block. This gives a cut $\tilde{x}$.
3. Bring the matrix X towards a good cut by using a convex-combination of X and $\tilde{x}\tilde{x}^T$. With this new matrix go to 1. and repeat as long as one finds better cuts.

It turned out, that with this heuristic for most of the instances the optimal cut was found at the root node of the Branch and Bound tree.

5 Random data for (MC) and (QP)

In this section some random data for presenting numerical results of our algorithm are specified. All the data sets can be downloaded from `http://www.math.uni-klu.ac.at/or/Software`. These instances are taken from various sources. Here we provide some of the characteristics of the data sets.

5.1 Max-Cut

5.1.1 Instances by the graph generator 'rudy'

The first group of instances follows [18] and consists of random graphs (of specified edge density) with various types of random edge weights. All graphs were produced by the graph generator 'rudy' [29]. For a detailed description and a list of the rudy-calls the reader is referred to the dissertation of Wiegele [30]. We generated ten instances of size $n = 100$ and given density d of the following types of graphs:

- $G_{0.5}$: unweighted graphs with density $d = 0.5$.
- $G_{-1/0/1}$: complete graphs with edge weights chosen uniformly from $\{-1, 0, 1\}$ and $d = 0.99$.
- $G_{[-10,10]}$: Graphs with integer edge weights chosen from $[-10, 10]$ and $d = 0.5$ and $d = 0.9$.
- $G_{[0,10]}$: Graphs with integer edge weights chosen from $[0, 10]$ and $d = 0.5$ and $d = 0.9$.

5.1.2 Applications in Statistical Physics: Ising instances

We also consider a set of test-problems of Frauke Liers [personal communication, 2005] coming from physical applications. The first group consists of two- and three-dimensional grid graphs with Gaussian distributed weights (zero mean and variance one). The second group consists of dense Ising instances which are obtained in the following way: all nodes lie evenly distributed on a cycle. The weights of the edges depend on the Euclidean distance between two nodes and a parameter σ, such that the proportion $c_{ij} \sim \frac{\epsilon_{ij}}{r_{ij}^{\sigma}}$ holds (ϵ_{ij} is chosen according to a Gaussian distribution with zero mean and variance one and r_{ij} is the Euclidean distance between nodes i and j).

5.2 Instances of (QP)

Pardalos and Rodgers [25] have proposed a test problem generator for unconstrained quadratic binary programming. Their routine generates a symmetric integer matrix Q to define the objective function for (QP), with the linear term c represented by the main diagonal of Q, and has several parameters to control the characteristics of the problem. These parameters are the number n of variables, the density d, i.e., the probability that a nonzero will occur in the off-diagonal part of Q, the lower and upper bounds of the main diagonal of Q are given by c^-, c^+. The lower and upper bounds for the off-diagonal part of Q are given by q^-, q^+. Furthermore we have $q_{ii} \sim$ discrete uniform in (c^-, c^+) and $q_{ij} = q_{ji} \sim$ discrete uniform in (q^-, q^+).

Several test problems generated this way are provided in the OR-library [4], [5]. We have chosen all the problems of sizes of our interest, which are the data sets `bqpgka`, due to [14] and `bqp100` and `bqp250`, see [6]. Furthermore, in [7] the sets `c` and `e` of `bqpgka` are extended. We call these instances `bqpbe`.

The characteristics are as follows:

- `bqpgka`:

	n	d	c^-	c^+	q^-	q^+
`bqpgka, set a`	$30, \ldots, 100$	$0.0625, \ldots, 0.5$	-100	100	-100	100
`bqpgka, set b`	$20, \ldots, 120$	1.0	0	63	-100	0
`bqpgka, set c`	$40, \ldots, 100$	$0.1, \ldots, 0.8$	-100	100	-50	50
`bqpgka, set d`	100	$0.1, \ldots, 1.0$	-75	75	-50	50
`bqpgka, set e`	200	$0.1, \ldots, 0.5$	-100	100	-50	50

- `beasley`: These instances have coefficients drawn from the interval $[-100, 100]$, i.e., $c^- = q^- = -100$ and $c^+ = q^+ = 100$, the density $d = 0.1$. There are 10 instances of dimension $n = 100$, called `beasley100` and 10 instances `beasley250` of dimension $n = 250$.

- `bqpbe`: Similar to the data sets `bqpgka c` and `e`, the lower and upper bounds are $c^- = -100, c^+ = 100$ and $q^- = -50, q^+ = 50$. The dimensions and densities are:

`bqpbe`	`100.1`	`120.3`	`120.8`	`150.3`	`150.8`	`200.3`	`200.8`	`250.1`
n	100	120	120	150	150	200	200	250
d	1.0	0.3	0.8	0.3	0.8	0.3	0.8	0.1

6 Numerical results

The algorithm was implemented in C and made publicly available for experimental runs as "Biq Mac" – a solver for *bi*nary *q*uadratic and *Max-C*ut problems at the site `http://biqmac.uni-klu.ac.at/`. If not stated otherwise, test runs were performed on a Pentium IV, 3.6 GHz and 2 GB RAM, operating system Linux. For a more detailed study of the numerical results the reader is referred to the dissertation [30].

6.1 Summarizing existing methods and their limits

Before we present our computational results, we summarize existing exact methods for (MC) together with their limits, as reported in the publications underlying these approaches.

LP: Linear programming based Branch and Bound approaches go back to Barahona et al. [3]. Liers et al. [23] enhance the algorithm and focus on solving toroidal grid graphs arising from physical applications, the so-called Ising model.

V: Linear programming combined with volume algorithm has been investigated by Barahona and Ladányi [1]. Also in this work, there is an emphasis on toroidal grid graphs.

EO: An exact approach using eigenvalue optimization based on (3) has been first investigated by Poljak and Rendl [27].

QP: The recent work of Billionnet and Elloumi [7] presents an approach based on convex quadratic optimization. This algorithm convexifies the objective function and uses a mixed-integer quadratic programming solver to obtain an exact solution of the problem.

SDPMET: An approach based on SDP and the triangle inequalities was first investigated by Helmberg and Rendl [18]. They solve (4) by an interior point algorithm.

SOCP: Kim and Kojima [21] and, later on, Muramatsu and Suzuki [24] use a second-order cone programming (SOCP) relaxation as bounding routine in a Branch and Bound framework to solve Max-Cut problems. However, the basic SDP relaxation performs better than their SOCP relaxation and the algorithm is capable of solving very sparse instances only. Therefore we omit comparing with this algorithm in the subsequent sections.

PP: Pardalos and Rodgers [25], [26] solve the quadratic program by Branch and Bound using a preprocessing phase where they try to fix some of the variables. The test on fixing the variables exploits information of the partial derivatives of the cost function.

In Table 1 we give a very naive overview of the capability of these approaches. We consider different types of instances and use the following symbols. A ✔ means, that the approach can solve instances of this type in a routine way. A ☕ indicates that one can have (at least) one cup of coffee while waiting for the solution and maybe there are instances that cannot be solved at all. The 🚲 suggests to have some holidays and come back in a couple of days to see whether the job is finished and the ☹ indicates that the chances for solving the problem with this method are very low. If we do not know, whether an algorithm can solve certain classes of instances or not, we indicate this with a question mark. Most likely, we could place ☹ instead of a question mark.

	LP	V	EO	QP	SDPMET	PP	Biq Mac
quadr 0-1, $n = 100, d = .1$	✔	✔	☹	✔	☕	✔	✔
quadr 0-1, $n = 250, d = .1$	?	?	☹	☹	☹	☹	☕
2-dim. torus, $n = 20 \times 20$	✔	✔	☹	☹	☹	?	🚲
3-dim. torus, $n = 7 \times 7 \times 7$	✔	✔	☹	☹	☹	?	☕
$G_{0.5}, n = 100$	☹	?	☹	🚲	🚲	?	✔
$G_{-1/0/1}, n = 100$	☹	?	🚲	🚲	🚲	?	✔

Table 1: Who can do what?

6.2 Max-Cut

6.2.1 Instances by the graph generator 'rudy'

Table 2 lists the computation times (minimum, average and maximum) and the number of nodes (minimum, average, maximum) of the resulting Branch and Bound (B&B) tree. The branching rule used for these kind of instances is R2.

graph	n	d	solved	min time (h:min)	avg time (h:min)	max time (h:min)	min nodes	avg nodes	max nodes
$G_{0.5}$	100	0.5	10	5	50	3:44	65	610	2925
$G_{-1/0/1}$	100	0.99	10	7	56	2:31	79	651	1811
$G_{[-10,10]}$	100	0.5	10	9	38	1:13	97	435	815
$G_{[-10,10]}$	100	0.9	10	5	57	3:12	51	679	2427
$G_{[1,10]}$	100	0.5	10	7	48	2:02	111	576	1465
$G_{[1,10]}$	100	0.9	10	12	40	1:26	155	464	1007

Table 2: Average Biq Mac results for Max-Cut problems. Run times on a Pentium IV, 3.6 GHz, 2GB RAM.

The average computation time for all instances is approximately one hour. Nevertheless, instances may also be solved within some minutes, and it could also take more than three hours for some graphs to obtain a solution.

The results show that on these classes of instances we outperform all other solution approaches known so far. The currently strongest results on these graphs are due to Billionnet and Elloumi [7]. They are not able to solve instances $G_{-1/0/1}$ of size $n = 100$ at all. Also, they could solve only two out of ten instances of $G_{0.5}$, $n = 100$.

6.2.2 Applications in Statistical Physics: Ising instances

As explained in Section 5.1.2, we consider two kinds of Ising instances: toroidal grid graphs and complete graphs.

Instances of the first kind can be solved efficiently by an LP-based Branch and Cut algorithm (see [23]). The computation times of their and our algorithm are reported in Table 3. As can be seen, on these sparse instances the LP-based method clearly outperforms our algorithm. However, we find a solution within a gap of 1% in reasonable time for all these samples.

The run-time of the Branch-Cut & Price algorithm [22] developed for the second kind of problems depends strongly on the parameter σ. For σ close to zero, we have a complete graph with Gaussian distributed weights. But for σ chosen suitably large, some of the edges become 'unimportant' and the pricing works very well for these graphs. In Table 4 the computation times of [22] and our algorithm are given. For $\sigma = 3.0$, we have roughly speaking the same computation times on the smallest instances. For the biggest ones, our approach clearly dominates. For $\sigma = 2.5$, the Branch-

Cut & Price algorithm already takes more than 20 hours for instances of size $n = 150$, whereas our algorithm needs almost similar computation times as in the $\sigma = 3.0$ case.

For both kinds of instances we used branching rule R3.

Problem number	n	[23] time	Biq Mac time	Problem number	n	[23] time	Biq Mac time
2 dimensional				3 dimensional			
g10_5555	100	0.15	10.12	g5_5555	125	2.68	18.01
g10_6666	100	0.14	15.94	g5_6666	125	3.29	24.52
g10_7777	100	0.18	14.89	g5_7777	125	3.07	26.00
g15_5555	225	0.44	304.03	g6_5555	216	20.56	280.85
g15_6666	225	0.78	359.87	g6_6666	216	37.74	2025.74
g15_7777	225	0.67	346.89	g6_7777	216	27.30	277.95
g20_5555	400	1.70	6690.99	g7_5555	343	95.25	432.71
g20_6666	400	3.50	35205.95	g7_6666	343	131.34	550.12
g20_7777	400	2.61	8092.80	g7_7777	343	460.01	117782.75

Table 3: Test runs on torus graphs with Gaussian distribution. Branch and Cut algorithm run on 1.8 GHz machine, Biq Mac done on a Pentium IV, 3.6 GHz. Time in seconds.

Problem number	n	[22] time	Biq Mac time	Problem number	n	[22] time	Biq Mac time
$\sigma = 3.0$				$\sigma = 2.5$			
100_5555	100	4:52	1:36	100_5555	100	18:22	1:32
100_6666	100	0:24	0:34	100_6666	100	6:27	1:06
100_7777	100	7:31	0:48	100_7777	100	10:08	0:47
150_5555	150	2:36:46	4:38	150_5555	150	21:28:39	4:25
150_6666	150	4:49:05	3:55	150_6666	150	23:35:11	5:39
150_7777	150	3:48:41	6:06	150_7777	150	31:40:07	9:19
200_5555	200	9:22:03	10:07	200_5555	200	–	10:05
200_6666	200	32:48:03	18:53	200_6666	200	–	17:55
200_7777	200	8:53:26	22:42	200_7777	200	–	21:38
250_5555	250	21:17:07	1:46:29	250_5555	250	–	3:00:28
250_6666	250	7:42:25	15:49	250_6666	250	–	1:17:04
250_7777	250	17:30:13	57:24	250_7777	250	–	1:10:50
300_5555	300	17:20:54	2:20:14	300_5555	300	–	6:43:47
300_6666	300	10:21:40	1:32:22	300_6666	300	–	9:04:38
300_7777	300	18:33:49	3:12:13	300_7777	300	–	13:00:10

Table 4: Test runs on Ising instances (complete graphs). Branch-Cut & Price on a 1.8 GHz Machine, Biq Mac on a 3.6 GHz PC. Times in hours:minutes:seconds.

6.3 Numerical results of (QP) instances

In this section we report the results for the instances derived from (QP). Best known lower and upper bounds for `bqpgka` and `beasley` data are reported at the pseudo-Boolean website [8]. Our results are as follows:

- `bqpgka`.
  - **Set a.** All problems are solved in the root node of the B&B tree within seconds.
  - **Set b.** These instances could all be solved, but were extremely challenging for our algorithm. The reason is, that the objective value in the Max-Cut formulation is of magnitude

10^6, and therefore even a relative gap of 0.1% does not allow to fathom the node. However, by allowing a relative error of at most 0.1%, we can solve all problems in the root node of the B&B tree.

- **Set c.** Similar to set a, also these instances were solved within a few seconds in the root node of the B&B tree.
- **Set d.** Here $n = 100$. The problems of set d could be solved within at most 7 minutes.
- **Set e.** We recall $n = 200$. The instances with densities 0.1, 0.2, 0.3 and 0.4 could all be solved within 2 hours of computation time. The instance with $d = 0.5$ has been solved after 35 hours. According to [8], none of these problems were solved before.

- `beasley`.
  Solving the 10 problems of size $n = 100$ can be done in the root node within one minute. Regarding the $n = 250$ instances, only two out of the ten problems have been solved before (see [8]), for the other eight problems we could prove optimality for the first time. Six out of these eight were solved within 5 hours, the other two needed 15 and 80 hours, respectively.
- `bqpbe`.
  We report the results of Billionnet and Elloumi [7] and our results in Table 5. As is shown in this table, [7] could not solve all out of the ten problems from the $n = 120$ variables and density 0.8 instances on, whereas our method still succeeded to solve them all. From the instances $n = 150, d = 0.8$ on, the convex-quadratic approach failed to solve any instance within their time limit of 3 hours. We still managed to obtain solutions to all of these instances (although for one graph it took about 54 hours to prove the optimality of the solution).

		[7]				Biq Mac			
			CPU time (sec)				CPU time (sec)		
n	d	solved	min	avg.	max	solved	min	avg.	max
100	1.0	10	27	372	1671	10	86	178	436
120	0.3	10	168	1263	4667	10	29	162	424
120	0.8	6	322	3909	9898	10	239	1320	3642
150	0.3	1		6789		10	1425	2263	2761
150	0.8	0		–		10	1654	1848	2133
200	0.3	0		–		10	7627	37265	193530
200	0.8	0		–		10	5541	47740	148515
250	0.1	0		–		10	12211	13295	16663

Table 5: Comparison between [7] and Biq Mac. Computation times of the convex-quadratic algorithm were obtained on a laptop Pentium IV, 1.6 GHz (time limit 3 hours), our results were computed on a Pentium IV of 3.6 GHz.

Deciding which branching rule is advisable for these instances is not so obvious anymore. Tentatively, for sparse problems R3 is superior, but the denser the instances are, the better is the performance of R2. A general recipe or an intelligent way of deciding at the top levels of the B&B tree which rule to follow would be very useful.

7 Equipartition

Finding a bisection of a graph such that each of the sets S and T have equal cardinality is often called *equipartition*. It is also customary to minimize the weight of edges in the cut. Hence the problem is a minor extension of (MC).

$$z_{EP} = \min\{x^T L x : e^T x = 0, x \in \{\pm 1\}^n\} \tag{8}$$

This leads to the following semidefinite relaxation.

$$z_{EP-SDP} = \min\{\mathrm{tr}LX : \mathrm{tr}JX = 0, \mathrm{diag}(X) = e, X \succeq 0\}, \tag{9}$$

where $J = ee^T$. Let A be the adjacency matrix of the given graph. We consider the Max-Cut instance with cost matrix $B = -A + J$. The "$-$" in $B = -A + J$ arises, because we minimize instead of maximizing, and the J comes from the constraint $\mathrm{tr}JX = 0$, that comes with a Lagrange multiplier (set equal to 1 for unweighted instances) into the objective function.

We consider the instances introduced in [19] of size $n = 124$ and $n = 250$ and summarize in Table 6 the best results for these instances known so far (see [20]). With our algorithm we prove optimality of the known lower bounds of all instances of size $n = 124$, and one of the instances of size $n = 250$. To the best of our knowledge, these exact solutions were obtained for the first time. The improved gap for the instances of size $n = 250$ and densities 0.02, 0.04 and 0.08 were obtained after a time limit of 32 hours cpu-time.

	best known					best known			
d	bound	$\lvert E_{cut}\rvert$	gap	new gap	d	bound	$\lvert E_{cut}\rvert$	gap	new gap
$n = 124$					$n = 250$				
0.02	12.01	13	0	0	0.01	26.06	29	2	0
0.04	61.22	63	1	0	0.02	103.61	114	10	8
0.08	170.93	178	7	0	0.04	327.88	357	29	22
0.16	440.08	449	8	0	0.08	779.55	828	48	35

Table 6: Best known results of the bisection problem for the Johnson graphs and the new gap obtained by Biq Mac.

8 Summary

In this paper we have presented an algorithm, that uses a Branch and Bound framework to solve the Max-Cut and related problems. At each node of the tree we calculate the bound by using a dynamic version of the bundle method that solves the basic semidefinite relaxation for Max-Cut strengthened by triangle inequalities. We conclude, that

- our approach solves any instance of all the test-bed considered with $n \approx 100$ nodes in a routine way. To the best of our knowledge, no other algorithm can manage these instances in a similar way.
- we solve problems of special structure and sparse problems up to $n = 300$ nodes.
- for the first time optimality could be proved for several problems of the OR-library. All problems that are reported at the Pseudo-Boolean website [8] with dimensions up to $n = 250$ are now solved.
- for the first time optimality of the bisection problem for some of the Johnson graphs has been proved, for those where we could not close the gap we reduced the best known gap significantly.
- for sparse problems it is not advisable to use our approach. Since linear programming based methods are capable of exploiting sparsity, solutions might be obtained much faster when applying these methods to sparse data.

Using our algorithm to solve this problem has been made publicly available at

`http://biqmac.uni-klu.ac.at/`.

References

[1] F. Barahona and L. Ladányi. Branch and cut based on the volume algorithm: Steiner trees in graphs and max-cut. *RAIRO Oper. Res.*, 40(1):53–73, 2006.

[2] 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, 1988.

[3] F. Barahona, M. Jünger, and G. Reinelt. Experiments in quadratic 0-1 programming. *Math. Programming*, 44(2, (Ser. A)):127–137, 1989.

[4] J. E. Beasley. Or-library: distributing test problems by electronic mail. *J. Oper. Res. Soc.*, 41 (11):1069–1072, 1990.

[5] J. E. Beasley. Or-library, 1990. `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`.

[6] J. E. Beasley. Heuristic algorithms for the unconstrained binary quadratic programming problem. Technical report, The Management School, Imperial College, London SW7 2AZ, England, 1998.

[7] A. Billionnet and S. Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Math. Programming*, 2006. to appear.

[8] E. Boros, P. L. Hammer, and G. Tavares. The pseudo-boolean optimization website, 2005. `http://rutcor.rutgers.edu/~pbo/`.

[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. *J. Statist. Phys.*, 80(1-2):487–496, 1995.

[10] C. Delorme and S. Poljak. Laplacian eigenvalues and the maximum cut problem. *Math. Programming*, 62(3, Ser. A):557–574, 1993.

[11] M. Elf, M. Jünger, and G. Rinaldi. Minimizing breaks by maximizing cuts. *Operations Research Letters*, 31:343–349, 2003.

[12] I. Fischer, G. Gruber, F. Rendl, and R. Sotirov. Computational experience with a bundle approach for semidefinite cutting plane relaxations of Max-Cut and equipartition. *Math. Programming*, 105(2-3, Ser. B):451–469, 2006.

[13] A. Frangioni, A. Lodi, and G. Rinaldi. New approaches for optimizing over the semimetric polytope. *Math. Program.*, 104(2-3, Ser. B):375–388, 2005.

[14] F. Glover, G. Kochenberger, and B. Alidaee. Adaptative memory tabu search for binary quadratic programs. *Management Sci.*, 44(3):336–345, 1998.

[15] M. X. Goemans and D. P. Williamson. .878-approximation algorithms for max cut and max 2sat. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 422–431, Montreal, Quebec, Canada, 1994.

[16] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6): 1115–1145, 1995. preliminary version see [15].

[17] C. Helmberg. Fixing variables in semidefinite relaxations. *SIAM J. Matrix Anal. Appl.*, 21(3): 952–969 (electronic), 2000.

[18] C. Helmberg and F. Rendl. Solving quadratic $(0,1)$-problems by semidefinite programs and cutting planes. *Math. Programming*, 82(3, Ser. A):291–315, 1998.

[19] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation. part i, graph partitioning. *Oper. Res.*, 37(6):865–892, 1989.

[20] S. E. Karisch and F. Rendl. Semidefinite programming and graph equipartition. In *Topics in semidefinite and interior-point methods (Toronto, ON, 1996)*, volume 18 of *Fields Inst. Commun.*, pages 77–95. Amer. Math. Soc., Providence, RI, 1998.

[21] S. Kim and M. Kojima. Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optim. Methods Softw.*, 15(3-4):201–224, 2001.

[22] F. Liers. *Contributions to Determining Exact Ground-States of Ising Spin-Glasses and to their Physics*. PhD thesis, Universität zu Köln, 2004.

[23] F. Liers, M. Jünger, G. Reinelt, and G. Rinaldi. Computing exact ground states of hard ising spin glass problems by branch-and-cut. In A. Hartmann and H. Rieger, editors, *New Optimization Algorithms in Physics*, pages 47–68. Wiley, 2004.

[24] M. Muramatsu and T. Suzuki. A new second-order cone programming relaxation for MAX-CUT problems. *J. Oper. Res. Soc. Japan*, 46(2):164–177, 2003.

[25] P. M. Pardalos and G. P. Rodgers. Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing*, 45(2):131–144, 1990.

[26] P. M. Pardalos and G. P. Rodgers. Parallel branch and bound algorithms for quadratic zero-one programs on the hypercube architecture. *Ann. Oper. Res.*, 22(1-4):271–292, 1990.

[27] S. Poljak and F. Rendl. Solving the max-cut problem using eigenvalues. *Discrete Appl. Math.*, 62(1-3):249–278, 1995.

[28] S. Poljak and F. Rendl. Nonpolyhedral relaxations of graph-bisection problems. *SIAM J. Optim.*, 5(3):467–487, 1995.

[29] G. Rinaldi. Rudy, 1998. `http://www-user.tu-chemnitz.de/~helmberg/rudy.tar.gz`.

[30] A. Wiegele. *Nonlinear optimization techniques applied to combinatorial optimization problems*. PhD thesis, Alpen-Adria-Universität Klagenfurt, 2006.