Combinatorial and Global Optimization, pp. 161–176
P.M. Pardalos, A. Migdalas and R. Burkard, Editors

Semidefinite Programming Approaches for MAX-2-SAT and MAX-3-SAT: computational perspectives

E. de Klerk (e.deklerk@twi.tudelft.nl)
Department of Technical Mathematics and Informatics,
Delft University of Technology,
Mekelweg 4, 2628 CD Delft, The Netherlands.

J.P. Warners (j.p.warners@twi.tudelft.nl)
Department of Technical Mathematics and Informatics,
Delft University of Technology,
Mekelweg 4, 2628 CD Delft, The Netherlands.

Abstract

Semidefinite programming (SDP) relaxations – in conjunction with randomized rounding schemes – yield 7/8 and 0.931 approximation algorithms for MAX-3-SAT and MAX-2-SAT respectively. In spite of these powerful theoretical results, it is not clear if SDP can be used as a practical tool for solving MAX-SAT problems to optimality. In this regard, the usefulness of the SDP approach will ultimately depend on the ability to exploit sparsity in the SDP relaxations of large dimension. (The dimension corresponds to the number of variables and the sparsity is related to the number of clauses.) We present an investigation of sparsity issues for the SDP relaxations of Goemans and Williamson [7], and Feige and Goemans [6] for MAX-2-SAT. Moreover, we test a branch and cut procedure to solve MAX-2-SAT to optimality, where the dual of the SDP relaxation is solved by interior point methods in order to exploit sparsity. The idea of exploiting sparsity in this way was first investigated for other combinatorial optimization problems by Benson, Ye, and Zhang [2]. Finally, based on this numerical experience we discuss possible extensions to MAX-3-SAT using the 7/8 relaxation of Karlow and Zwick [12] for MAX-3-SAT.

Keywords: Semidefinite programming, satisfiability, interior point algorithms

1 Introduction

An instance of the MAX-SAT problem is defined by a collection of Boolean clauses $\{C_1, \ldots, C_k\}$, where each clause is a disjunction of literals drawn from a set of variables $\{x_1, \ldots, x_n\}$. A literal is either a variable x_i or its negation $\neg x_i$ for some i. Each clause has an associated nonnegative weight, and an optimal solution to a MAX-SAT instance is an assignment of truth values to the variables which maximizes the total weight of the satisfied clauses. MAX-p-SAT is a special case of MAX-SAT where each clause contains at most p literals.

There are many algorithms available for the MAX-SAT problem, even though the MAX-2-SAT problem is already NP-complete. Algorithms for MAX-SAT include well-known procedures such as EDPL, and branch and cut; approximation algorithms include GRASP, GSAT, and recently semidefinite programming (SDP) (see e.g. [10] and the references therein). SDP became a powerful tool for MAX-2-SAT problems when Goemans and Williamson [7] proved that a 0.87856 approximation algorithm could be obtained by rewriting the MAX-2-SAT problem as a Boolean quadratic programming problem, and solving a convex SDP relaxation of the resulting problem, followed by a randomized rounding procedure. This rounding procedure may be seen as a heuristic which gives a solution with a quality guarantee, while the optimal value of the SDP gives a bound on the sub-optimality of this heuristic solution. If the heuristic solution is not shown to be optimal by the bound, then the relaxation must be tightened by adding suitable cuts. Recent numerical studies indicate that it is very hard to prove optimality for MAX-2-SAT by only tightening the SDP relaxation [10]. It seems necessary therefore to use the SDP relaxations in some branch and cut framework. Up to now, the bottleneck for such an approach has been that it is hard to exploit sparsity in the solution procedure of the SDP relaxations. In this paper, we show that one can solve the dual of the SDP relaxations efficiently, using the dual interior point method of Benson, Ye and Zhang [2]. Moreover, we show that the relaxed solutions can be used in a branch and cut scheme to solve MAX-2-SAT problems with 50 variables and up to 500 clauses to optimality in a few minutes on a workstation.

2 The SDP relaxation of MAX-2-SAT

The key in the SDP reformulation is to introduce new Boolean variables $(y_1, \ldots, y_n) \in \{-1, 1\}^n$ and to express the number of unsatisfied clauses as a quadratic function of these new variables.

Note that a clause $x_i \vee x_j$ is satisfied if $y_i + y_j \geq 0$, etc. Thus we can represent the satisfiability problem as a feasibility problem: find $y \in \{-1, 1\}^n$ so that $Ay \geq 0$,

where A is a suitable $k \times n$ matrix.

Note that

$$(Ay)_i - 1 = \left\{ \begin{array}{cc} 1 \text{ or } -1 & \text{if clause } i \text{ is satisfied} \\ -3 & \text{otherwise} \end{array} \right\} \tag{1}$$

Letting e denote the vector of all-ones, it follows that

$$(Ay - e)^T(Ay - e) = 9unsat + (k - unsat)$$

where *unsat* denotes the number of unsatisfied clauses for the assignment y of truth values. Using $e^Te = k$, we have that

$$unsat = \frac{1}{8}\left(y^TA^TAy - 2e^TAy\right),$$

and the MAX-2-SAT problem is simply to minimize this quantity over $y \in \{-1,1\}^n$. Thus we have rewritten MAX-2-SAT as a Boolean quadratic programming problem:

$$unsat^* := \min_y \left\{ \frac{1}{8}\left(y^TA^TAy - 2e^TAy\right) \mid y \in \{-1,1\}^n \right\}.$$

Such problems have standard semidefinite relaxations with provable quality bounds [16]. The first step in deriving the relaxation is to remove the linear term in the objective by adding an additional Boolean variable $y_{n+1} \in \{-1,1\}$ to obtain

$$unsat^* = \min_y \left\{ \frac{1}{8}\left(y^TA^TAy - 2y_{n+1}e^TAy\right) \mid (y, y_{n+1}) \in \{-1,1\}^{n+1} \right\}. \tag{2}$$

Note that the introduction of the auxiliary variable y_{n+1} does not change the optimal value of the optimization problem. Problem (2) can be further simplified by using the structure of A: Note that one has

$$\sum_{i=1}^{n}(A^TA)_{ii} = 2k,$$

which shows that

$$unsat^* = \min_y \left\{ \frac{1}{8}\left(y^T\left[A^TA - \mathrm{diag}\,(A^TA)\right]y - 2y_{n+1}e^TAy + 2k\right) \mid (y, y_{n+1}) \in \{-1,1\}^{n+1} \right\}$$

We can rewrite this as:

$$\frac{1}{4}k + \min_{\tilde{y}} \left\{ \tilde{y}^TW\tilde{y} \;:\; \tilde{y} \in \{-1,1\}^{n+1} \right\}, \tag{3}$$

where $\tilde{y} := [y_1, \ldots, y_n, y_{n+1}]$ and W is the $(n+1) \times (n+1)$ matrix:

$$W := \frac{1}{8}\left[\begin{array}{cc} A^TA - \mathrm{diag}\,(A^TA) & -A^Te \\ -e^TA & 0 \end{array} \right]$$

Note that W_{ij} can only be nonzero if x_i and x_j appear together in some clause. Thus the fraction of nonzeros of W will never exceed the ratio $(k+n) : \frac{1}{2}n(n+1)$. For example, for a MAX-2-SAT instance with $n = 100$ variables and $k = 400$ clauses the upper bound on the density of W is 9.8%.

We will show in later sections why this ratio is an important consideration when choosing an algorithm for solving the SDP relaxation.

The SDP relaxation of (3) can be derived by rewriting it as

$$unsat^* := \frac{1}{4}k + \min_{\tilde{y}} \left\{ \mathbf{Tr}\left(W\tilde{y}\tilde{y}^T\right) \ : \ \mathrm{diag}\left[\tilde{y}\tilde{y}^T\right] = e \right\},$$

where '**Tr**' denotes the trace operator. The SDP relaxation is now obtained by replacing the rank one, positive semidefinite matrix $\tilde{y}\tilde{y}^T$ by a positive semidefinite matrix X which is not restricted to have rank one.

The relaxation therefore takes the form:

$$unsat^* \geq SDP^* := \frac{1}{4}k + \min_{X} \left\{ \mathbf{Tr}\left(WX\right) \ : \ \mathrm{diag}\left(X\right) = e, \ X \succeq 0 \right\}, \tag{4}$$

where $X \succeq 0$ means X is symmetric positive semidefinite. Note that all products y_iy_j are replaced by the matrix entries X_{ij}. Given a Choleski decomposition of X, say $X = V^TV$, one can write $X_{ij} = (v^i)^Tv^j$ where the v^i's are the columns of V. This means that the product y_iy_j is in fact relaxed to an inner product $(v^i)^Tv^j$.

This type of relaxation was originally suggested by Lovasz and Schrijver [13]. Goemans and Williamson [7] proved that

$$\frac{k - unsat^*}{k - SDP^*} \geq 0.87856. \tag{5}$$

(We should note that the derivation of the model presented here is different from that in [7], and follows that of Van Maaren and Warners [14].)

Also note that $W_{ij} = 0 \ \forall (i,j)$ if all possible clauses are included for the n variables. In this case exactly $\frac{3}{4}$ of the clauses are satisfiable, i.e. $unsat^* = \frac{1}{4}k$. In this case one trivially has $SDP^* = unsat^*$, i.e. the SDP relaxation is exact. It is therefore reasonable to expect that the SDP relaxation will become even tighter as the ratio k/n grows. This can be observed from numerical experiments shown in Table 1, where the average ratio for the left hand side of expression (5) is given as a function of the number of clauses (for random MAX-2-SAT instances with 50 variables).

Goemans and Williamson also proposed the following heuristic for use in conjunction with the SDP relaxation:

- Solve the SDP relaxation (4) to obtain an ϵ-optimal $X = V^TV$.

# clauses	$\frac{k-unsat^*}{k-SDP^*}$
50	0.95658
100	0.97502
200	0.98538
500	0.98911
1000	0.99128
2000	0.99258
3000	0.99400
4000	0.99503
5000	0.99592

Table 1: The average quality of the SDP relaxation improves as the number of clauses grows (50 variables).

- Choose a random $r \in \mathbb{R}^n$ and normalize r.
- Set $y_i = 1$ if $r^T v^i \geq 0$ or set $y_i = -1$ otherwise.

This randomized algorithm yields an approximate solution to MAX-2-SAT with expected objective value at least 0.87 times the optimal value.

3 Additional valid inequalities

There exist instances of MAX-2-SAT where the ratio in (5) is no better than 0.88889 [6]. The SDP relaxation (4) can be strengthened by adding a number of valid inequalities (cuts) proposed by Feige and Goemans [6].

The first set of inequalities (called *triangle inequalities*) are based on the observation that for any pair of indices (i, j) there holds:

$$\begin{aligned} y_{n+1}y_i + y_{n+1}y_j + y_iy_j &\geq -1 \\ -y_{n+1}y_i - y_{n+1}y_j + y_iy_j &\geq -1 \\ -y_{n+1}y_i + y_{n+1}y_j - y_iy_j &\geq -1. \end{aligned}$$

In the SDP relaxation these inequalities correspond to $\frac{3}{2}n(n-1)$ additional linear constraints:

$$\left.\begin{array}{rcl} X_{n+1,i} + X_{n+1,j} + X_{ij} & \geq & -1 \\ -X_{n+1,i} - X_{n+1,j} + X_{ij} & \geq & -1 \\ -X_{n+1,i} + X_{n+1,j} - X_{ij} & \geq & -1. \end{array}\right\} \tag{6}$$

Note that the first of these constraints can be rewritten as

$$\frac{1}{2}\mathbf{Tr}\left(e_{i,j,n+1}e_{i,j,n+1}^T X\right) - 1\frac{1}{2} \geq -1$$

where the vector $e_{i,j,n+1}$ has ones in the positions i, j and $n+1$, and zero elsewhere. In other words, we have a constraint of the form

$$\mathbf{Tr}\,(A_i X) \geq 1, \tag{7}$$

where A_i is a rank one matrix containing only zero's and ± 1. The other two constraints in (6) can be represented similarly. Feige and Goemans have shown that the addition of these inequalities improves the quality guarantee of the SDP relaxation from 0.87856 to 0.93109 (see (5)). A bound on the worst-case approximation is 0.94513, i.e. there exist problems where the ratio (5) is no larger than 0.94513.

The constraints (6) form a subset of an even larger set of $\frac{2}{3}n(n-1)(n-2)$ valid inequalities which follow from:

$$\begin{array}{rcl} y_k y_i + y_k y_j + y_i y_j & \geq & -1 \\ y_k y_i - y_k y_j - y_i y_j & \geq & -1 \\ -y_k y_i + y_k y_j - y_i y_j & \geq & -1 \\ -y_k y_i - y_k y_j + y_i y_j & \geq & -1 \end{array}$$

for each distinct triple of indices (i, j, k). The corresponding constraints in the SDP relaxation once again takes the form (7). The quality guarantee for the additional cuts remains 0.93109, but the worst-known behaviour now becomes 0.98462.

In practice all these inequalities cannot be added beforehand because of the increase in problem size; it is more feasible to re-solve the SDP relaxation after having added (some of) the violated inequalities.

4 Solving the SDP relaxation of MAX-2-SAT

The SDP relaxations mentioned so far can be cast in the generic form

$$\min_X \mathbf{Tr}\,(WX)$$

subject to

$$\begin{aligned} \operatorname{diag}(X) &= e \\ \mathbf{Tr}(A_i X) &\geq 1, \quad i = 1, \ldots, m \\ X &\succeq 0, \end{aligned}$$

where the A_i's are rank one matrices corresponding to the valid inequalities in Section 3.

The associated dual problem is

$$\max_{\gamma, y, S} \sum_{i=1}^{m} y_i + \sum_{i=1}^{n} \gamma_i$$

subject to

$$\begin{aligned} D(\gamma) + \sum_{i=1}^{m} y_i A_i + S &= W \\ y \geq 0, \gamma \in \mathbb{R}^n, S \succeq 0, & \end{aligned}$$

where $D(\gamma)$ denotes the diagonal matrix with the vector $\gamma \in \mathbb{R}^n$ on its diagonal.

Note that the dual matrix S will have more or less the same sparsity structure as W, if the number of cuts m is small. Recall further that W will be sparse in general, as discussed in Section 2.

This suggests to solve the dual problem instead of the primal in order to exploit this sparsity structure.

Dual interior point methods are based on the dual logarithmic barrier function

$$f_b(S, y) = \log \det(S) + \sum_{i=1}^{n} \log(y_i),$$

which can be added to the dual objective function in order to replace the (matrix) inequality constraints $S \succeq 0$ and $y \geq 0$. Thus one can solve a sequence of problems of the form

$$\max_{\gamma, y, S} \left\{ \sum_{i=1}^{m} y_i + \sum_{i=1}^{n} \gamma_i + \mu f_b(S, y) \right\} \tag{8}$$

subject to

$$D(\gamma) + \sum_{i=1}^{m} y_i A_i + S = W,$$

for decreasing values of $\mu > 0$. The projected Newton direction for this problem can be calculated from a positive definite linear system with coefficient matrix consisting

of four blocks (see the collected works [5, 1, 8, 4, 2]):

$$M := \begin{bmatrix} S^{-1} \circ S^{-1} & B \\ B^T & C \end{bmatrix}$$

where '$\circ$' indicates the Hadamard (componentwise) product, and the blocks B and C are respectively of the form

$$B_{ij} = \mathbf{Tr}\left(A_i S^{-1} e_j e_j^T S^{-1}\right) = e_j^T S^{-1} A_i S^{-1} e_j$$

where e_j is the jth standard unit vector, and

$$C_{ij} = \mathbf{Tr}\left(A_i S^{-1} A_j S^{-1}\right).$$

Once S^{-1} is known, the computation of $[S^{-1} \circ S^{-1}]_{ij}$, B_{ij} and C_{ij} all require only one multiplication and some additions.

The matrix M can therefore be assembled quickly once the inverse S^{-1} has been computed. Detail of how to compute S^{-1} efficiently in general is given by Benson, Ye, and Zhang in [2]. They have implemented a dual scaling method (using the search direction described above[1]) which requires $O(\sqrt{m+n})$ iterations for convergence. The computation per iteration is dominated by the solution of the linear system with coefficient matrix M. This algorithm is used in the numerical experiments below.

To give an impression of how fast the relaxed problem can be solved using this approach, the CPU-times (in seconds) for MAX-2-SAT relaxations of some of the largest benchmark problems from [9] is given in Table 2. The computation was done on a HP 9000/715 workstation.

The column 'solution' gives the best obtained solution for the Goemans Williamson heuristic, upper gives the bound $k - SDP^*$, and 'ratio' indicates the ratio of the best obtained heuristic solution to $k - SDP^*$.

5 A branch and cut framework

The SDP relaxations can be used in any branch and cut framework. The framework we have used for our numerical experiments will be described in this section, with reference to Figure 1.

[1]The dual scaling method chooses the parameter μ in (8) dynamically by monitoring the progress of the algorithm via the so-called Tanabe-Todd-Ye potential function. For details the reader is referred to [2].

problem	clauses	variables	solution	upper	ratio	time sdp	time heuristic
p2300-1	300	99	285	292	0.9762	4.09	0.32
p2300-2	300	101	285	294	0.9710	4.08	0.35
p2300-3	300	101	287	293	0.9808	3.91	0.34
p2300-4	300	101	286	293	0.9772	4.02	0.35
p2300-5	300	101	284	293	0.9694	4.90	0.34
p2300-6	300	101	283	290	0.9774	4.38	0.35
p2300-7	300	101	279	288	0.9709	4.74	0.36
p2300-8	300	101	283	291	0.9757	4.11	0.36
p2300-9	300	101	284	291	0.9761	4.26	0.34
p2400-1	400	101	369	379	0.9761	5.26	0.40
p2400-2	400	101	371	380	0.9780	5.13	0.41
p2400-3	400	101	373	383	0.9764	4.37	0.39
p2400-4	400	101	371	378	0.9827	4.65	0.40
p2400-5	400	101	370	379	0.9780	5.22	0.39
p2400-6	400	101	372	380	0.9796	4.82	0.41
p2400-7	400	101	373	382	0.9788	4.44	0.40
p2400-8	400	101	366	376	0.9748	4.77	0.40

Table 2: Solution times for the SDP relaxation of MAX-2-SAT and for the Goemans-Williamson heuristic.

At any node in the branching tree, the current set of clauses (obtained after partial assignment of the variables) is denoted by Φ, and *lb* and *ub* contain lower and upper bounds on the minimal number of unsatisfiable clauses respectively. The value *unsat* is a counter for the number of unsatisfied clauses by the current partial assignment. Note that *lb* and *unsat* are *local* variables that are only valid in the current branch; on the other hand, *ub* is a global variable which is valid for the whole search tree. At termination of the procedure, *ub* contains the optimal value of the instance.

Before calling **node_procedure** the values *lb*, *unsat*, and *ub* must be initialized. One can take $lb := 0$, $unsat := 0$, $ub := k$. Following Borchers and Furman, unit resolution is applied if $ub - unsat = 1$. Subsequently, the semidefinite relaxation of the current

```
procedure node_procedure (Φ, lb, unsat);
   if (ub − unsat = 1) unit_resolution(Φ);
   (lb_sdp, ub_sdp):= SDP_relaxation(Φ);                    (*)
   ub := min{ub, unsat + ub_sdp};                           (*)
   lb := max{lb, unsat + max{0, lb_sdp}};                   (*)
   if (ub − lb < 1) return;
   x :=branch_rule(Φ);
   Set x ← TRUE and update Φ, unsat;
   if (ub − unsat ≥ 1) node_procedure (Φ, lb, unsat);
   Set x ← FALSE and update Φ, unsat;
   if (ub − unsat ≥ 1) node_procedure (Φ, lb, unsat);
return;
```

Figure 1: Branch and cut framework for MAX-2-SAT

formula is solved to obtain upper and lower bounds ub_{sdp} and lb_{sdp}. The current bounds ub and lb are then updated (taking *unsat* into account), and if $ub - lb < 1$, then the best known solution so far cannot be improved upon in the current branch, so that we backtrack. Otherwise a variable x is determined to branch on, which is set to TRUE and FALSE respectively. The branching rule for fixing variables is as follows: choose the variable with the *highest occurrence in the longest clauses.*

The formula Φ and *unsat* are subsequently updated; if $ub - unsat < 1$ this branch need not be further explored. In the other case `node_procedure` is recursively called.

In each node, the steps marked (*) can be repeated adding violated cuts from Section 3 to the relaxation, to obtain tighter bounds.

6 Numerical experiments

In this section we present some numerical results for the branch and cut SDP algorithm of the previous section. The results presented here are of a preliminary nature, and were obtained *without* adding extra cuts.

The MAX-2-SAT benchmark problems are taken from Borchers and Furman [3] (except for the two largest instances).

As before, all reported CPU-times are in seconds on a HP-9000/715 workstation with 160MB internal memory.

The SDP branch and cut method presented here are compared to a modified EDPL algorithm [3] and to the Mixed integer linear programming approach using the commercial solver Minto [15]. The respective CPU-times are shown in Table 3.

It is immediately clear that the SDP approach is distinctly superior to the other two approaches if the clauses/variables ratio exceeds 4:1. The reason is that the SDP relaxation becomes tighter as this ratio grows, as discussed in Section 2. Note also that the SDP branch and cut algorithm solved each of the problems in a few minutes. It therefore has a very robust performance in comparison to the other two methods.

The second set of test problems are *weighted* MAX-2-SAT problems from Borchers and Furman [3]. The same observations hold as for the unweighted problems, although the difference is now somewhat less pronounced. All the algorithms fare somewhat better on these problems. The results are shown in Table 4.

# clauses	SDP (nodes)	EDPL	Minto
100	84 (82)	1.36	12.9
150	69 (64)	5.1	18.0
200	91 (70)	395	67.3
250	118 (92)	2218	128
300	170 (128)	29794	687
350	127 (91)	>12hr	2339
400	56 (40)	>12hr	1550
450	276 (210)	>12hr	12634
500	205 (144)	>12hr	8677
2500	331 (184)	not run	not run
5000	663 (399)	not run	not run

Table 3: Solution times (in seconds) of MAX-2-SAT benchmark problems ($n = 50$) for different algorithms

7 Future work

7.1 Cutting planes

The results from the previous section for MAX-2-SAT can probably be improved upon significantly by adding some of the cuts described in Section 3 to the relaxations.

The influence of added cuts is illustrated in Table 5. These results were obtained using the SDP solver CUTSDP [11] in the branching scheme described in Section 5. The solution times are for proving optimality only, and are given for two MAX-2-SAT instances from Table 3 and two from Table 4 (weighted). The solver CUTSDP uses a primal-dual predictor-corrector algorithm based on the so-called XS direction. This direction also results in a sparse Newton system at each iteration of the solution of MAX-2-SAT relaxation, but the algorithm still requires additional computation involving the dense primal matrix variable. For this reason, it is not as efficient as the dual scaling method. However, the CUTSDP software automatically adds (some of) the violated triangle inequalities described in Section 3, and therefore gives and indication of the effect of cutting planes on the branching procedure.

# clauses	SDP (nodes)	EDPL	Minto
100	101 (125)	1.36	12.9
150	101 (108)	2.04	16.3
200	58 (61)	23.5	34.1
250	137 (117)	235	171
300	61 (44)	874	149
350	161 (122)	40285	2155
400	100 (82)	20233	579
450	53 (44)	>12hr	1420
500	118 (76)	>12hr	3153

Table 4: Solution times (in seconds) for weighted MAX-2-SAT benchmark problems ($n = 50$) for different algorithms

# clauses	CUTSDP with cuts	(nodes)	CUTSDP without cuts	(nodes)
100	283	(25)	206	(78)
450	396	(38)	403	(191)
100 (weighted)	180	(20)	228	(116)
450 (weighted)	270	(28)	80	(40)

Table 5: Solution times (in seconds) for MAX-2-SAT benchmark problems ($n = 50$) for the CUTSDP method (with and without cuts) in a branching framework

It is clear from Table 5 that the introduction of cuts reduces the number of nodes in the branching tree significantly, but increases the solution time of the relaxations at the nodes. The total solution time is not improved in general, and all the solution times are worse than those reported in Table 3 and Table 4 for the dual scaling method without cuts.

Nevertheless, it is clear that the number of branching nodes can be reduced significantly; the challenge is therefore to extend the dual scaling method to use cuts and to

find the optimal trade-off between stronger relaxations and increased solution times.

7.2 Extension to MAX-3-SAT

Another challenging problem is to extend the approach in this paper to MAX-3-SAT problems. The are two possibilities in this regard:

(1) One can rewrite MAX-3-SAT as a MAX-2-SAT problem in such a way that the SDP relaxation to the MAX-2-SAT problem yields a 0.801 approximation algorithm for MAX-3-SAT (see [17]); the resulting MAX-2-SAT problem can then be solved using the approach in this paper.

(2) One can use the branch and cut formulation of this paper in conjunction with the MAX-3-SAT relaxation of Karlov and Zwick [12]. This relaxation guarantees a 7/8 approximation.

The difficulty with approach (1) is that one has seven 2-clauses for each clause of MAX-3-SAT. For example, the clause $a \vee b \vee c$ is replaced by the seven (weighted) clauses

$$2a \vee z,\ \neg b \vee z,\ b \vee \neg z,\ \neg c \vee z,\ c \vee \neg z,\ b \vee c,\ \neg b \vee \neg c,$$

where z is an auxiliary variable. If the MAX-3-SAT instance therefore has k clauses and n variables, the associated MAX-2-SAT instance has $7k$ clauses and $n+k$ variables. The MAX-2-SAT instance is also highly structured, and it remains to be seen if the SDP relaxations are as effective in this case as for random instances.

The approach (2) involves the following relaxation of MAX-3-SAT: the clause $x_i \vee x_j \vee x_k$ will be true if and only if

$$\min \left\{ \begin{array}{ll} 1 - \frac{1}{4}(x_{n+1} + x_i)(x_j + x_k), & 1 - \frac{1}{4}(x_{n+1} + x_j)(x_i + x_k) \\ 1 - \frac{1}{4}(x_{n+1} + x_k)(x_i + x_j), & 1 \end{array} \right\} = 1.$$

We can relax the left hand side to four linear matrix inequalities by replacing x_i by a vector v_i of norm one, etc., and replacing the products by inner products, as before:

$$\begin{aligned} t &\leq 1 - \frac{1}{4}(v_{n+1} + v_i)^T(v_j + v_k) \\ t &\leq 1 - \frac{1}{4}(v_{n+1} + v_j)^T(v_i + v_k) \\ t &\leq 1 - \frac{1}{4}(v_{n+1} + v_k)^T(v_i + v_j) \\ t &\leq 1. \end{aligned}$$

The SDP relaxation involves maximizing the sum of the values t for all the clauses. It is easy to check that one obtains an SDP with $4k$ inequality constraints where the coefficient matrices of the constraints are of rank 3.

One can still solve the resulting problem by the dual scaling method (see [2]), but the assembly of the linear system at each iteration becomes more expensive, and its coefficient matrix becomes more dense. The question is therefore if these relaxations can be solved quickly enough to allow incorporation in a branch and cut scheme.

References

[1] K.M. Anstreicher and M. Fampa. A long-step path following algorithm for semidefinite programming problems. Working Paper, Department of Management Sciences, University of Iowa, Iowa City, USA, 1996.

[2] S.J. Benson, Y. Ye, and X. Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. Working paper, Computational Optimization LAb, Dept. of Management Science, University of Iowa, Iowa City, USA, 1997.

[3] B. Borchers and J. Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. Manuscript, 1997. (To appear in *Journal of Combinatorial Optimization*).

[4] E. de Klerk. *Interior Point Methods for Semidefinite Programming.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 1997.

[5] L. Faybusovich. Semi–definite programming: a path–following algorithm for a linear–quadratic functional. *SIAM Journal on Optimization*, 6(4):1007–1024, 1996.

[6] U. Feige and M. Goemans. Approximating the value of two prover proof systems with applications to MAX 2SAT and MAX DICUT. In *Proc. Third Israel Symposium on Theory of Computing and Systems*, 1995.

[7] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[8] B. He, E. de Klerk, C. Roos, and T. Terlaky. Method of approximate centers for semi–definite programming. *Optimization Methods and Software*, 7:291–309, 1997.

[9] S. Joy, J. Mitchell, and B. Borchers. A branch and cut algorithm for MAX–SAT and weighted MAX–SAT. In M.A. Trick and D.S. Johnson, editors, *DIMACS series in Discrete Mathematics and Theoretical Computer Science*, volume 26. American Mathematical Society.

[10] S. Joy, J. Mitchell, and B. Borchers. Solving max-sat and weighted max-sat problems using branch-and-cut. Manuscript, 1998.

[11] S. Karisch. CUTSDP – A toolbox for a cutting-plane approach based on semidefinite programming. User's Guide/Version 1.0. Technical Report IMM-REP-1998-10, Dept. Mathematical Modelling, Technical University of Denmark, 1998.

[12] H. Karlow and U. Zwick. A 7/8-approximation algorithm for max 3sat? Manuscript, 1997.

[13] L. Lovász and A. Schrijver. Cones of matrices and set–functions and 0–1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

[14] H. Van Maaren and J.P. Warners. Bounds and fast approximation algorithms for binary quadratic optimization problems with application to MAX 2SAT and MAX CUT. Technical Report 97–35, Delft University of Technology, The Netherlands, 1997.

[15] G.L. Nemhauser, M.W.P. Savelsbergh, and G.C. Sigismondi. Minto, a mixed integer optimizer. *Operations Research Letters*, 15(1):47–58, 1994.

[16] Yu. Nesterov. Quality of semidefinite relaxation for nonconvex quadratic optimization. CORE Discussion paper 9719, Belgium, March 1997.

[17] L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation and linear programming. In *Proc. of the 37th Anual IEEE Symposium on Foundations of Computer Science*. 1996.