Math Meth Oper Res (2004) 60:349–367
DOI 10.1007/s001860400377

Mathematical Methods
of Operations Research


# On semidefinite programming relaxations for the satisfiability problem

**Miguel F. Anjos**

Operational Research Group, School of Mathematics, University of Southampton, Southampton, SO17 1BJ, United Kingdom. (e-mail: anjos@stanfordalumni.org)

Manuscript received: June 2003/ Final version received: March 2004

**Abstract.** This paper is concerned with the analysis and comparison of semidefinite programming (SDP) relaxations for the satisfiability (SAT) problem. Our presentation is focussed on the special case of 3-SAT, but the ideas presented can in principle be extended to any instance of SAT specified by a set of boolean variables and a propositional formula in conjunctive normal form. We propose a new SDP relaxation for 3-SAT and prove some of its theoretical properties. We also show that, together with two SDP relaxations previously proposed in the literature, the new relaxation completes a trio of linearly sized relaxations with increasing rank-based guarantees for proving satisfiability. A comparison of the relative practical performances of the SDP relaxations shows that, among these three relaxations, the new relaxation provides the best tradeoff between theoretical strength and practical performance within an enumerative algorithm.

**Key words:** Satisfiability problem, Semidefinite programming, Combinatorial optimization, Global optimization

## 1. Introduction

The satisfiability (SAT) problem is a central problem in mathematical logic, computing theory, and artificial intelligence. We consider instances of SAT specified by a set of boolean variables and a propositional formula in conjunctive normal form. Given such an instance, the SAT problem asks whether there is a truth assignment to the variables such that the formula is satisfied. It is well known that SAT is in general NP-complete, although several important special cases can be solved in polynomial time. There has been great interest in the design of algorithms to solve the SAT problem; see [17] for an extensive survey.

Semidefinite programming (SDP) refers to the class of optimization problems where a linear function of a matrix variable $X$ is maximized (or minimized) subject to linear constraints on the elements of $X$ and the additional constraint that $X$ must be positive semidefinite. This includes linear programming problems as a special case, namely when all the matrices involved are diagonal. A variety of polynomial-time interior-point algorithms for solving SDPs have been proposed in the literature, and several excellent solvers for SDP are now available. We refer the reader to the SDP webpage [19] as well as the handbook [37] for a thorough coverage of the theory and algorithms in this area, as well as several application areas where semidefinite programming researchers have made significant contributions.

We note that SDP has been successfully applied in the development of approximation algorithms for hard combinatorial optimization problems. The survey paper [28] provides an excellent overview of the results in this area. In particular, SDP is the basis for recent approximation algorithms for a related class of problems, the MAX-$k$-SAT problems. (The notation $k$-SAT refers to the instances of SAT for which all the clauses have length at most $k$.) Given a set of propositional clauses all of length at most $k$ in conjunctive normal form, the MAX-$k$-SAT problem consists of determining the largest number of clauses that can be satisfied simultaneously by any given truth assignment. The seminal paper of Goemans and Williamson [15] proposed and analyzed an approximation algorithm using SDP for the MAX-2-SAT problem. The approximation algorithm of Karloff and Zwick for MAX-3-SAT [22] was shown to be optimal (unless $P = NP$), and further extensions have been proposed by Zwick [38] and Halperin and Zwick [18]. These results provide the best known approximation guarantees for MAX-$k$-SAT problems.

We are interested in the application of SDP to the basic SAT problem, and in particular in how SDP can be used to prove unsatisfiability. In [12, 11], de Klerk, van Maaren, and Warners introduced an SDP relaxation for SAT, the so-called Gap relaxation. They show that the Gap relaxation characterizes unsatisfiability for some interesting classes of SAT problems, such as mutilated chessboard and pigeonhole instances, as well as for 2-SAT. (Note that unlike MAX-2-SAT, 2-SAT is solvable in polynomial-time [5].) However, it cannot detect unsatisfiability when all the clauses have length three or higher.

More recently, we introduced in [2] an improved SDP relaxation which can be used to prove that a given SAT formula is unsatisfiable, independently of the lengths of the clauses in the instance. This relaxation is easily defined for every instance of SAT, and it inherits all the favourable properties of the Gap relaxation. It is constructed using ideas from a "higher liftings" paradigm for constructing SDP relaxations of discrete optimization problems. The use of liftings has been proposed in the literature within the framework of general purpose lift-and-project methods for 0-1 optimization [6,33,30] and the study of SDP relaxations for specific 0-1 problems dates back at least to Lovász's introduction of the so-called theta function as a bound for the stability number of a graph [29]. Alternatively, if a 0-1 problem is viewed as a feasibility problem over an algebraic set, then SDP relaxations can be obtained using Hilbert's positivstellensatz [32]. For further recent research on liftings for discrete optimization problems, see [8,16].

The lifting paradigm we consider can be summarized as follows. Suppose that we have a discrete optimization problem on $n$ binary variables. The SDP

relaxation in the space of $(n+1) \times (n+1)$ symmetric matrices is called a first lifting. Note that, except for the first row, the rows and columns of the matrix variable in this relaxation are indexed by the binary variables themselves. To generalize this operation, we allow the rows and columns of the SDP relaxations to be indexed by *subsets* of the discrete variables in the formulation. These larger matrices can be interpreted as higher liftings, in the spirit of the second lifting proposed by Anjos and Wolkowicz [4], and its generalization independently proposed by Lasserre [25, 26].

Considering these higher liftings, an interesting question is to find conditions on the rank of an optimal matrix for the SDP relaxation which ensure that the optimal value of the SDP is actually the optimal value of the underlying discrete problem. For liftings of the Maximum-Cut (max-cut) problem, the rank-1 case is obvious since the optimal solution of the SDP is then a vertex of the cut polytope (see e.g. [13]). For second liftings, a rank-2 guarantee of optimality was proved by Anjos and Wolkowicz [3], and this result was extended to the whole of the Lasserre hierarchy by Laurent[27]. From a theoretical point of view, these rank-based conditions for optimality can be interpreted as a measure of the relative strength of the relaxations. From a practical point of view, they are helpful because of the occurrence of SDP solutions with high rank when there are multiple optimal solutions to the original discrete problem. This happens because interior-point algorithms typically converge to a matrix in the interior of the optimal face of the cone of positive semidefinite matrices, and in the presence of multiple solutions this face contains matrices of ranks higher than one. Therefore, the ability to detect optimality for as high a rank value as possible will often allow an enumerative algorithm to avoid further branching steps and potentially yield a significant reduction in computational time.

This paper is concerned with the practical application of the higher liftings to discrete optimization problems, with a particular focus on SAT. Practical computation with these liftings has proved difficult. The computational results reported in [1], where second liftings for max-cut problems with only up to 27 binary variables were successfully solved, motivated us to consider "partial" liftings which are more amenable to practical computation than the complete higher liftings, while preserving as far as possible the desirable properties of the complete liftings. One positive step in this direction was taken in [2], where the dimensions of the proposed SDP relaxation for SAT depend linearly on the number of clauses in the SAT instance, yet the relaxation has a rank-3 guarantee for proving satisfiability. This means that if the rank of the optimal solution of the SDP is 1, 2, or 3, then we can extract a satisfying truth assignment from that optimal solution, and hence obtain a certificate of satisfiability. The higher rank value guaranteeing a proof of satisfiability thus reflects a greater ability to prove satisfiability, although it also implies that a greater computational effort is required to solve the SDP. The computational results reported in [2] show that the SDP relaxation used in conjunction with an enumerative search procedure is still impractical for solving SAT problems with more than, say, 100 clauses, unless the solution is obtained without resorting to branching. Branching is not always necessary, and in particular this relaxation successfully proved the unsatisfiability of some hard instances that remained unsolved in the SAT 2003 competition, showing that the SDP approach has the potential to complement existing SAT algorithms.

The results in this paper improve the applicability of SDP to SAT. We shall focus on 3-SAT, although the ideas we present can in principle be extended to any instance of satisfiability. The main contribution of this paper is a new SDP relaxation for 3-SAT which is more compact than that proposed in [2], but which is nonetheless able to detect both satisfiability and unsatisfiability of 3-SAT instances. Indeed, it is straightforward to prove that if the SDP relaxation is infeasible, then the given 3-SAT instance is unsatisfiable. Furthermore, if a feasible matrix $Y$ is found, and its rank equals 1 or 2, then it yields a truth assignment that satisfies the SAT instance, thus proving satisfiability of the instance. In this sense, we say that this new relaxation is endowed with a rank-2 guarantee for proving satisfiability. Therefore, together with the aforementioned relaxations, it completes a trio of linearly sized relaxations of increasing dimension and with correspondingly increasing rank-based guarantees for proving satisfiability. We present computational results showing that a basic enumerative algorithm using this relaxation is able to routinely prove either satisfiability or unsatisfiability of Uniform Random-3-SAT instances with more than 300 clauses from the DIMACS set of benchmark problems. The computational results also suggest that the new relaxation is more effective than previous relaxations in the literature when using an SDP-based enumerative approach for solving instances of SAT in conjunctive normal form. Therefore, this new relaxation is another step towards a practical SDP-based algorithm for satisfiability.

This paper is structured as follows. In the next section we introduce some notation, give a formal definition of the SAT and 3-SAT problems, and present some previous work in the literature. In Section 3 we present the construction of the new SDP relaxation, and in Section 4 we state and prove some theoretical properties of this relaxation. In Section 5, we show how this new relaxation completes a trio of linearly sized SDP relaxations for SAT. Section 6 contains a brief explanation of the algorithm we implemented to compare the practical performance of the three relaxations. Finally, Section 7 summarizes some directions for future research.

## 2. Formulation and previous SDP relaxations for SAT

We consider the SAT problem for instances in conjunctive normal form (CNF). Such instances are specified by a set of variables $x_1, \ldots, x_n$ and a propositional formula $\Phi = \bigwedge_{j=1}^{m} C_j$, with each clause $C_j$ having the form $C_j = \bigvee_{k \in I_j} x_k \vee \bigvee_{k \in \bar{I}_j} \bar{x}_k$ where $I_j, \bar{I}_j \subseteq \{1, \ldots, n\}$, $I_j \cap \bar{I}_j = \emptyset$, and $\bar{x}_i$ denotes the negation of $x_i$. (We assume without loss of generality that $|I_j \cup \bar{I}_j| \geq 2$ for every clause $C_j$.) The SAT problem is: Given a satisfiability instance, is $\Phi$ satisfiable, that is, is there a truth assignment to the variables $x_1, \ldots, x_n$ such that $\Phi$ evaluates to TRUE?

We shall henceforth let TRUE be denoted by 1 and FALSE be denoted by $-1$. For clause $j$ and $k \in I_j \cup \bar{I}_j$, define

$$s_{j,k} := \begin{cases} 1, & \text{if} \quad k \in I_j \\ -1, & \text{if} \quad k \in \bar{I}_j \end{cases} \tag{1}$$

The SAT problem is now equivalent to the integer programming feasibility problem

$$\begin{aligned}&\text{find } x \in \{\pm 1\}^n\\ &\text{s.t. } \sum_{k \in I_j \cup \bar{I}_j} s_{j,k} x_k \geq 2 - l(C_j), \quad j = 1, \ldots, m\end{aligned}$$

where $l(C_j) = |I_j \cup \bar{I}_j|$ denotes the number of literals in clause $C_j$. Clearly this problem is equivalent to the original SAT problem, and hence is in general NP-complete. Some special cases of SAT can be solved in polynomial time using linear programming, see [10]. Special instances of SAT with certain constraints on the length of the clauses are often of particular interest, both theoretically and in practice; we refer the reader to the survey [17]. In this paper, we focus on 3-SAT, which refers to those instances of SAT for which each clause $C_j$ satisfies $|I_j \cup \bar{I}_j| \leq 3$. Nonetheless, the ideas in this paper extend in a straightforward manner to any instance of satisfiability.

The initial study of the application of SDP to SAT was done by de Klerk, van Maaren, and Warners who introduced the Gap relaxation for SAT [12, 11]. The Gap relaxation for 3-SAT may be expressed as follows:

$$\begin{aligned}&\text{find } X \in \mathcal{S}^{n+1}\\ &\text{s.t.}\\ &\qquad s_{j,i_1} s_{j,i_2} X_{i_1,i_2} - s_{j,i_1} X_{0,i_1} - s_{j,i_2} X_{0,i_2} + 1 = 0,\\ &\qquad\qquad \text{where } \{i_1, i_2\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 2\\ &(R_1)\ s_{j,i_1} s_{j,i_2} X_{i_1,i_2} + s_{j,i_1} s_{j,i_3} X_{i_1,i_3} + s_{j,i_2} s_{j,i_3} X_{i_2,i_3} - s_{j,i_1} X_{0,i_1}\\ &\qquad\qquad - s_{j,i_2} X_{0,i_2} - s_{j,i_3} X_{0,i_3} \leq 0,\\ &\qquad\qquad \text{where } \{i_1, i_2, i_3\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 3\\ &\qquad\qquad\qquad \text{diag}(X) = e\\ &\qquad\qquad\qquad X \succeq 0\end{aligned}$$

where $\mathcal{S}^n$ denotes the space of $n \times n$ square symmetric matrices, $\text{diag}(X)$ represents a vector containing the diagonal elements of the matrix $X$, $e$ denotes the vector of all ones, and $X \succeq 0$ denotes that $X$ is positive semidefinite. This SDP relaxation is based on the elliptic approximations of clauses introduced in [35] and it characterizes unsatisfiability for 2-SAT problems. (It is well known that 2-SAT is solvable in polynomial time [5].) For later reference, we state this result as a theorem:

**Theorem 1.** *[12, Theorem 5.1] For every instance of 2-SAT, the corresponding Gap relaxation is feasible if and only if the instance is satisfiable.* □

More interestingly, the Gap relaxation also characterizes satisfiability for certain classes of SAT problems, such as mutilated chessboard and pigeonhole instances. Rounding schemes and approximation guarantees for the Gap relaxation, as well as its behaviour on so-called $(2+p)$-SAT problems, are studied in [11]. Finally, we note that the Gap relaxation is always feasible when the instance has no clauses of length less than three, and hence is unable to detect unsatisfiability for such instances.

More recently, the author proposed an improved SDP relaxation which is able to detect unsatisfiability independently of the length of the clauses, and inherits all the properties of the Gap relaxation. The general construction and analysis of this relaxation are presented in [2]. We outline here the derivation of that relaxation for the specific case of 3-SAT, which is the focus of this paper. In doing so, we set the stage for the introduction of the new SDP

relaxation in Section 3. With the parameters $s_{j,k}$ as defined above, Proposition 1 in [2] implies that:

– If $l(C_j) = 2$ and $\{i_1, i_2\} = I_j \cup \bar{I}_j$, then $C_j$ is satisfied by $x_{i_1}, x_{i_2} \in \{\pm 1\}$ if and only if
$$s_{j,1}x_{i_1} + s_{j,2}x_{i_2} - s_{j,1}s_{j,2}x_{i_1}x_{i_2} = 1.$$

– If $l(C_j) = 3$ and $\{i_1, i_2, i_3\} = I_j \cup \bar{I}_j$, then $C_j$ is satisfied by $x_{i_1}, x_{i_2}, x_{i_3} \in \{\pm 1\}$ if and only if
$$s_{j,1}x_{i_1} + s_{j,2}x_{i_2} + s_{j,3}x_{i_3} - s_{j,1}s_{j,2}x_{i_1}x_{i_2} - s_{j,1}s_{j,3}x_{i_1}x_{i_3} - s_{j,2}s_{j,3}x_{i_2}x_{i_3} + s_{j,1}s_{j,2}s_{j,3}x_{i_1}x_{i_2}x_{i_3} = 1.$$

Therefore, we can formulate the 3-SAT problem as follows:

$$\begin{aligned}
&\text{find } x_1 \ldots, x_n \\
&\text{s.t.} \\
&\quad s_{j,1}x_{i_1} + s_{j,2}x_{i_2} - s_{j,1}s_{j,2}x_{i_1}x_{i_2} = 1, \quad \text{where } \{i_1, i_2\} = I_j \cup \bar{I}_j, \quad \text{if } l(C_j) = 2 \\
&\quad s_{j,1}x_{i_1} + s_{j,2}x_{i_2} + s_{j,3}x_{i_3} - s_{j,1}s_{j,2}x_{i_1}x_{i_2} - s_{j,1}s_{j,3}x_{i_1}x_{i_3} - s_{j,2}s_{j,3}x_{i_2}x_{i_3} \\
&\qquad + s_{j,1}s_{j,2}s_{j,3}x_{i_1}x_{i_2}x_{i_3} = 1, \text{ where } \{i_1, i_2, i_3\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 3 \\
&\quad x_i^2 = 1, \quad i = 1, \ldots, n
\end{aligned}$$

The next step consists of formulating the problem in symmetric matrix space. Let $\mathcal{P}$ denote the set of all nonempty sets $I \subseteq \{1, \ldots, n\}$ such that the term $\prod_{i \in I} x_i$ appears in the above formulation. Also introduce new variables:
$$x_I := \prod_{i \in I} x_i$$
for each $I \in \mathcal{P}$, define the vector
$$v := (1, x_{I_1}, \ldots, x_{I_{|\mathcal{P}|}})^T,$$
and define the square symmetric positive semidefinite rank-one matrix
$$Y := vv^T$$
whose rows and columns are indexed by $\emptyset \cup \mathcal{P}$. By construction of the matrix variable $Y$, we have that $Y_{\emptyset,I} = x_I$ for all $I \in \mathcal{P}$. Using these new variables, we can formulate 3-SAT as:

$$\begin{aligned}
&\text{find } Y \in \mathcal{S}^{|\mathcal{P}|+1} \\
&\text{s.t.} \\
&\quad s_{j,1}Y_{\emptyset,\{i_1\}} + s_{j,2}Y_{\emptyset,\{i_2\}} - s_{j,1}s_{j,2}x_{i_1}x_{i_2}Y_{\emptyset,\{i_1,i_2\}} = 1, \\
&\quad \text{where } \{i_1, i_2\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 2 \\
&\quad s_{j,1}Y_{\emptyset,\{i_1\}} + s_{j,2}Y_{\emptyset,\{i_2\}} + s_{j,3}Y_{\emptyset,\{i_3\}} - s_{j,1}s_{j,2}Y_{\emptyset,\{i_1,i_2\}} - s_{j,1}s_{j,3}Y_{\emptyset,\{i_1,i_3\}} \\
&\qquad - s_{j,2}s_{j,3}Y_{\emptyset,\{i_2,i_3\}} + s_{j,1}s_{j,2}s_{j,3}Y_{\emptyset,\{i_1,i_2,i_3\}} = 1, \\
&\quad \text{where } \{i_1, i_2, i_3\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 3 \\
&\quad \text{diag}(Y) = e \\
&\quad \text{rank}(Y) = 1 \\
&\quad Y \succeq 0
\end{aligned}$$

This formulation has the form of an SDP, except for the requirement that $\text{rank}(Y) = 1$. In order to improve the SDP relaxation obtained by removing the rank constraint, we now observe that $\text{rank}(Y) = 1$ implies that for every

triple $I_1, I_2, I_3$ of subsets of indices in $\mathcal{P}$ such that the symmetric difference of any two equals the third, the following three equations hold:

$$Y_{\emptyset,I_1} = Y_{I_2,I_3}, \quad Y_{\emptyset,I_2} = Y_{I_1,I_3}, \quad \text{and} \quad Y_{\emptyset,I_3} = Y_{I_1,I_2}. \tag{2}$$

Therefore we can add some or all of these constraints to the formulation above without invalidating it. We add to it the equations of the form (2) for all the triples $\{I_1, I_2, I_3\} \subseteq \mathcal{P}$ such that $(I_1 \cup I_2 \cup I_3) \subseteq (I_j \cup \bar{I}_j)$ for some clause $j$ and satisfying the symmetric difference condition. Although these constraints are redundant when the rank constraint is enforced, they will make a difference in the SDP obtained when the rank restriction is removed. Indeed, in general we can require that

$$Y_{I_1,I_2} = Y_{I_3,I_4} \text{ whenever } I_1 \Delta I_2 = I_3 \Delta I_4 \tag{3}$$

where $I_i \Delta I_j$ denotes the symmetric difference of $I_i$ and $I_j$. The tradeoff involved in the choice of additional constraints is that as the number of constraints increases, the semidefinite relaxations become computationally more expensive to solve. The motivation for the particular choice of redundant constraints in (2) is that they suffice to prove Theorem 2 below, a corollary of the main result in [2]. Thus we obtain the SDP relaxation for 3-SAT:

$$(R_3) \qquad \begin{array}{l} \text{find } Y \in \mathcal{S}^{|\mathcal{P}|+1} \\ \text{s.t.} \\ \quad s_{j,1} Y_{\emptyset,\{i_1\}} + s_{j,2} Y_{\emptyset,\{i_2\}} - s_{j,1} s_{j,2} x_{i_1} x_{i_2} Y_{\emptyset,\{i_1,i_2\}} = 1, \\ \qquad \text{where } \{i_1, i_2\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 2 \\ \quad s_{j,1} Y_{\emptyset,\{i_1\}} + s_{j,2} Y_{\emptyset,\{i_2\}} + s_{j,3} Y_{\emptyset,\{i_3\}} - s_{j,1} s_{j,2} Y_{\emptyset,\{i_1,i_2\}} - s_{j,1} s_{j,3} Y_{\emptyset,\{i_1,i_3\}} \\ \qquad - s_{j,2} s_{j,3} Y_{\emptyset,\{i_2,i_3\}} + s_{j,1} s_{j,2} s_{j,3} Y_{\emptyset,\{i_1,i_2,i_3\}} = 1, \\ \qquad \text{where } \{i_1, i_2, i_3\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 3 \\ \quad Y_{\emptyset,I_1} = Y_{I_2,I_3}, \quad Y_{\emptyset,I_2} = Y_{I_1,I_3}, \quad \text{and} \quad Y_{\emptyset,I_3} = Y_{I_1,I_2}, \quad \forall \{I_1, I_2, I_3\} \subseteq P \\ \qquad \text{such that } I_1 \Delta I_2 = I_3 \text{ and } (I_1 \cup I_2 \cup I_3) \subseteq (I_j \cup \bar{I}_j) \text{ for some } j \\ \qquad \text{diag}(Y) = e \\ \qquad Y \succeq 0 \end{array}$$

We refer to this relaxation as $R_3$, and it has the following properties:

**Theorem 2.** *[2, Theorem 2] Given a 3-SAT propositional formula in CNF, the following statements hold for the semidefinite relaxation* $R_3$:

- *If* $R_3$ *is infeasible, then the formula is unsatisfiable.*
- *If* $R_3$ *is feasible, and* $Y$ *is a feasible matrix such that* rank $Y \leq 3$, *then a truth assignment satisfying the formula can be obtained from* $Y$. *Hence the formula is satisfiable.*

The computational properties of this relaxation were also studied in [2], where it was successfully used to prove satisfiability and unsatisfiability of 3-SAT instances with 50 variables and up to 100 clauses.

The results in this paper improve the practical applicability of SDP to solve general 3-SAT instances. In Section 3, we propose a compact semidefinite relaxation $R_2$ that is linearly sized with respect to the number of clauses, improves on $R_1$ (the Gap relaxation), and is computationally superior to $R_3$ thanks to significant reductions in the dimension of the matrix variable and in the number of linear constraints. The matrix variable of the compact

SDP relaxation can be viewed as a principal submatrix of the matrix variable in $R_3$. We show that, compared with $R_3$, the proposed semidefinite relaxation $R_2$ retains the ability to prove unsatisfiability, and that although it does not retain the rank-3 guarantee, it has a rank-2 guarantee. Hence, it is a compromise relaxation between the Gap and $R_3$, and it completes a trio of linearly sized semidefinite relaxations with correspondingly stronger rank guarantees. Although we focus on 3-SAT, we note that the idea behind the construction in Section 3 extends in a straightforward manner to general instances of SAT.

## 3. Construction of the compact SDP relaxation

We begin with the formulation for 3-SAT:

$$\begin{aligned}
&\text{find } x_1 \ldots, x_n \\
&\text{s.t.} \\
&\quad s_{j,1}x_{i_1} + s_{j,2}x_{i_2} - s_{j,1}s_{j,2}x_{i_1}x_{i_2} = 1, \quad \text{where } \{i_1, i_2\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 2 \\
&\quad s_{j,1}x_{i_1} + s_{j,2}x_{i_2} + s_{j,3}x_{i_3} - s_{j,1}s_{j,2}x_{i_1}x_{i_2} - s_{j,1}s_{j,3}x_{i_1}x_{i_3} - s_{j,2}s_{j,3}x_{i_2}x_{i_3} \\
&\quad + s_{j,1}s_{j,2}s_{j,3}x_{i_1}x_{i_2}x_{i_3} = 1, \text{ where } \{i_1, i_2, i_3\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 3 \\
&\quad x_i^2 = 1, \quad i = 1, \ldots, n.
\end{aligned}$$

We then construct a different formulation in matrix space by choosing a different collection of column indices. In [2], the set $\mathcal{P}$ of column indices contained all the nonempty sets $I \subseteq (I_j \cup \bar{I}_j)$ corresponding to products of $x_i$ variables appearing in the above formulation. To obtain a more compact SDP relaxation, we choose a smaller set of column indices, namely

$$\mathcal{O} := \{I \mid I \subseteq (I_j \cup \bar{I}_j) \text{ for some } j, |I| \bmod 2 = 1\}.$$

The set $\mathcal{O}$ is a strict subset of the set $\mathcal{P}$, consisting of the sets of odd cardinality in $\mathcal{P}$. It is clear that the sets in $\mathcal{P}$ of even cardinality corresponding to terms appearing in the above formulation are all generated as symmetric differences of the sets in $\mathcal{O}$.

Having chosen our set of column indices, we introduce new variables

$$x_I := \prod_{i \in I} x_i,$$

for each $I \in \mathcal{O}$, define the vector

$$v := (1, x_{I_1}, \ldots, x_{I_{|\mathcal{O}|}})^T,$$

and the rank-one matrix

$$Y := vv^T,$$

whose rows and columns are indexed by $\emptyset \cup \mathcal{O}$. By construction of $Y$, we have $Y_{\emptyset,I} = x_I$ for all $I \in \mathcal{O}$ and $Y_{\{\min(I)\},I\Delta\{\min(I)\}} = x_I$ for all $I \in \mathcal{P}\backslash\mathcal{O}$. (Note that $T\Delta\{\min(T)\}$ is an element of $\mathcal{P}\backslash\mathcal{O}$ when $|T|$ is even.)

This means that the new variables corresponding to subsets of logical variables of odd cardinality appear exactly once in the first row of $Y$, and the new variables corresponding to subsets of even cardinality have the "representative" matrix entries $Y_{\{\min(T)\},T\Delta\{\min(T)\}}$. Therefore, we can formulate the SAT problem as:

$$
\begin{aligned}
&\text{find } Y \in \mathcal{S}^{|\mathcal{O}|+1}\\
&\text{s.t.}\\
&\qquad s_{j,i_1} Y(\{i_1\}) + s_{j,i_2} Y(\{i_2\}) - s_{j,i_1} s_{j,i_2} Y(\{i_1,i_2\}) = 1,\\
&\qquad \text{where } \{i_1,i_2\} = I_j \cup \bar{I}_j, \quad \text{if } l(C_j) = 2\\
&\qquad s_{j,i_1} Y(\{i_1\}) + s_{j,i_2} Y(\{i_2\}) + s_{j,i_3} Y(\{i_3\}) - s_{j,i_1} s_{j,i_2} Y(\{i_1,i_2\})\\
&\qquad -s_{j,i_1} s_{j,i_3} Y(\{i_1,i_3\}) - s_{j,i_2} s_{j,i_3} Y(\{i_2,i_3\}) + s_{j,i_1} s_{j,i_2} s_{j,i_3} Y(\{i_1,i_2,i_3\}) = 1,\\
&\qquad \text{where } \{i_1,i_2,i_3\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 3\\
&\qquad \operatorname{diag}(Y) = e\\
&\qquad \operatorname{rank}(Y) = 1\\
&\qquad Y \succeq 0
\end{aligned}
$$

where

$$
Y(T) = \begin{cases} Y_{\emptyset,T}, & |T| \text{ odd;} \\ Y_{\{\min(T)\},T\Delta\{\min(T)\}}, & |T| \text{ even.} \end{cases}
$$

Now we want to improve the semidefinite relaxations that we will obtain by adding some constraints of the type in (3). We choose to add, for each clause $C_j$ such that $l(C_j) = 3$ with $\{i_1,i_2,i_3\} = I_j \cup \bar{I}_j$, the constraints

$$
Y_{\{i_1\},\{i_2\}} = Y_{\{i_3\},\{i_1,i_2,i_3\}}, Y_{\{i_1\},\{i_3\}} = Y_{\{i_2\},\{i_1,i_2,i_3\}}, Y_{\{i_2\},\{i_3\}} = Y_{\{i_1\},\{i_1,i_2,i_3\}}. \tag{4}
$$

The motivation for the particular choice of constraints in (4) is that they suffice to prove Theorem 3 below.

Finally, omitting the rank constraint gives us the $R_2$ relaxation:

$$
\begin{aligned}
&\text{find } Y \in \mathcal{S}^{|\mathcal{O}|+1}\\
&\text{s.t.}\\
&\qquad s_{j,i_1} Y(\{i_1\}) + s_{j,i_2} Y(\{i_2\}) - s_{j,i_1} s_{j,i_2} Y(\{i_1,i_2\}) = 1,\\
&\qquad \text{where} \{i_1,i_2\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 2\\
&\qquad s_{j,i_1} Y(\{i_1\}) + s_{j,i_2} Y(\{i_2\}) + s_{j,i_3} Y(\{i_3\}) - s_{j,i_1} s_{j,i_2} Y(\{i_1,i_2\})\\
(R_2) \qquad &\qquad -s_{j,i_1} s_{j,i_3} Y(\{i_1,i_3\}) - s_{j,i_2} s_{j,i_3} Y(\{i_2,i_3\})\\
&\qquad + s_{j,i_1} s_{j,i_2} s_{j,i_3} Y(\{i_1,i_2,i_3\}) = 1, \text{ where } \{i_1,i_2,i_3\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 3\\
&\qquad Y_{\{i_1\},\{i_2\}} = Y_{\{i_3\},\{i_1,i_2,i_3\}}, \quad Y_{\{i_1\},\{i_3\}} = Y_{\{i_2\},\{i_1,i_2,i_3\}},\\
&\qquad \text{and } Y_{\{i_2\},\{i_3\}} = Y_{\{i_1\},\{i_1,i_2,i_3\}} \text{ where } \{i_1,i_2,i_3\} = I_j \cup \bar{I}_j, \text{ if } l(C_j) = 3\\
&\qquad \operatorname{diag}(Y) = e\\
&\qquad Y \succeq 0
\end{aligned}
$$

Note that for 2-SAT this relaxation is precisely the $R_1$ relaxation.

We illustrate this construction with an example.

**Example 1.** Suppose we are given the CNF formula

$$
(x_1 \vee x_2) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (x_3 \vee \bar{x}_4 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_6)
$$

Then

$$
\mathcal{O} = \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{234\},\{345\},\{126\}\},
$$

and the matrix variable $Y$, if it is rank-one, has the form

$$\begin{pmatrix}
1 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_2x_3x_4 & x_3x_4x_5 & x_1x_2x_6 \\
x_1 & 1 & x_1x_2 & x_1x_3 & x_1x_4 & x_1x_5 & x_1x_6 & x_1x_2x_3x_4 & x_1x_3x_4x_5 & x_2x_6 \\
x_2 & x_1x_2 & 1 & x_2x_3 & x_2x_4 & x_2x_5 & x_2x_6 & x_3x_4 & x_2x_3x_4x_5 & x_1x_6 \\
x_3 & x_1x_3 & x_2x_3 & 1 & x_3x_4 & x_3x_5 & x_3x_6 & x_2x_4 & x_4x_5 & x_1x_2x_3x_6 \\
x_4 & x_1x_4 & x_2x_4 & x_3x_4 & 1 & x_4x_5 & x_4x_6 & x_2x_3 & x_3x_5 & x_1x_2x_4x_6 \\
x_5 & x_1x_5 & x_2x_5 & x_3x_5 & x_4x_5 & 1 & x_5x_6 & x_2x_3x_4x_5 & x_3x_4 & x_1x_2x_5x_6 \\
x_6 & x_1x_6 & x_2x_6 & x_3x_6 & x_4x_6 & x_5x_6 & 1 & x_2x_3x_4x_6 & x_3x_4x_5x_6 & x_1x_2 \\
x_2x_3x_4 & x_1x_2x_3x_4 & x_3x_4 & x_2x_4 & x_2x_3 & x_2x_3x_4x_5 & x_2x_3x_4x_6 & 1 & x_2x_5 & x_1x_3x_4x_6 \\
x_3x_4x_5 & x_1x_3x_4x_5 & x_2x_3x_4x_5 & x_4x_5 & x_3x_5 & x_3x_4 & x_3x_4x_5x_6 & x_2x_5 & 1 & x_1x_2x_3x_4x_5x_6 \\
x_1x_2x_6 & x_2x_6 & x_1x_6 & x_1x_2x_3x_6 & x_1x_2x_4x_6 & x_1x_2x_5x_6 & x_1x_2 & x_1x_3x_4x_6 & x_1x_2x_3x_4x_5x_6 & 1
\end{pmatrix}$$

Using the construction presented, we have the matrix variable $Y$ partially patterned on the structure of the rank-one matrix as follows:

$$Y = \begin{pmatrix}
1 & Y_{\emptyset,\{1\}} & Y_{\emptyset,\{2\}} & Y_{\emptyset,\{3\}} & Y_{\emptyset,\{4\}} & Y_{\emptyset,\{5\}} & Y_{\emptyset,\{6\}} & Y_{\emptyset,\{2,3,4\}} & Y_{\emptyset,\{3,4,5\}} & Y_{\emptyset,\{1,2,6\}} \\
 & 1 & Y_{\{1\},\{2\}} & * & * & * & Y_{\{1\},\{6\}} & * & * & Y_{\{2\},\{6\}} \\
 & & 1 & Y_{\{2\},\{3\}} & Y_{\{2\},\{4\}} & * & Y_{\{2\},\{6\}} & Y_{\{3\},\{4\}} & * & Y_{\{1\},\{6\}} \\
 & & & 1 & Y_{\{3\},\{4\}} & Y_{\{3\},\{5\}} & * & Y_{\{2\},\{4\}} & Y_{\{4\},\{5\}} & * \\
 & & & & 1 & Y_{\{4\},\{5\}} & * & Y_{\{2\},\{3\}} & Y_{\{3\},\{5\}} & * \\
 & & & & & 1 & * & * & Y_{\{3\},\{4\}} & * \\
 & & & & & & 1 & * & * & Y_{\{1\},\{2\}} \\
 & & & & & & & 1 & * & * \\
 & & & & & & & & 1 & * \\
 & & & & & & & & & 1
\end{pmatrix}$$

where the lower triangle of $Y$ is fixed by symmetry. The elements of $Y$ denoted by asterisks are not involved in any of the linear equality constraints, although they are of course constrained by the positive semidefiniteness constraint.

We then express the conditions for satisfiability using the entries in $Y$. For each clause in the formula, we need one equality constraint:

$$\begin{aligned}
(x_1 \vee x_2) &\Rightarrow \quad Y_{\emptyset,\{1\}} + Y_{\emptyset,\{2\}} - Y_{\{1\},\{2\}} = 1; \\
(x_2 \vee \bar{x}_3 \vee x_4) &\Rightarrow Y_{\emptyset,\{2\}} - Y_{\emptyset,\{3\}} + Y_{\emptyset,\{4\}} + Y_{\{2\},\{3\}} - Y_{\{2\},\{4\}} + Y_{\{3\},\{4\}} - Y_{\emptyset,\{2,3,4\}} = 1; \\
(x_3 \vee \bar{x}_4 \vee \bar{x}_5) &\Rightarrow Y_{\emptyset,\{3\}} - Y_{\emptyset,\{4\}} - Y_{\emptyset,\{5\}} + Y_{\{3\},\{4\}} + Y_{\{3\},\{5\}} - Y_{\{4\},\{5\}} + Y_{\emptyset,\{3,4,5\}} = 1; \\
(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_6) &\Rightarrow - Y_{\emptyset,\{1\}} - Y_{\emptyset,\{2\}} - Y_{\emptyset,\{6\}} - Y_{\{1\},\{2\}} - Y_{\{1\},\{6\}} - Y_{\{2\},\{6\}} - Y_{\emptyset,\{1,2,6\}} = 1;
\end{aligned}$$

and thus the $R_2$ relaxation is

$$\begin{aligned}
&\text{find } Y \in \mathcal{S}^{10} \\
&\text{s.t.} \\
&\qquad Y_{\emptyset,\{1\}} + Y_{\emptyset,\{2\}} - Y_{\{1\},\{2\}} = 1 \\
&Y_{\emptyset,\{2\}} - Y_{\emptyset,\{3\}} + Y_{\emptyset,\{4\}} + Y_{\{2\},\{3\}} - Y_{\{2\},\{4\}} + Y_{\{3\},\{4\}} - Y_{\emptyset,\{2,3,4\}} = 1 \\
&Y_{\emptyset,\{3\}} - Y_{\emptyset,\{4\}} - Y_{\emptyset,\{5\}} + Y_{\{3\},\{4\}} + Y_{\{3\},\{5\}} - Y_{\{4\},\{5\}} + Y_{\emptyset,\{3,4,5\}} = 1 \\
&-Y_{\emptyset,\{1\}} - Y_{\emptyset,\{2\}} - Y_{\emptyset,\{6\}} - Y_{\{1\},\{2\}} - Y_{\{1\},\{6\}} - Y_{\{2\},\{6\}} - Y_{\emptyset,\{1,2,6\}} = 1 \\
&Y_{\{2\},\{6\}} = Y_{\{1\},\{1,2,6\}} \\
&Y_{\{3\},\{4\}} = Y_{\{2\},\{2,3,4\}} \\
&Y_{\{1\},\{6\}} = Y_{\{2\},\{1,2,6\}} \\
&Y_{\{2\},\{4\}} = Y_{\{3\},\{2,3,4\}} \\
&Y_{\{4\},\{5\}} = Y_{\{3\},\{3,4,5\}} \\
&Y_{\{2\},\{3\}} = Y_{\{4\},\{2,3,4\}} \\
&Y_{\{3\},\{5\}} = Y_{\{4\},\{3,4,5\}} \\
&Y_{\{3\},\{4\}} = Y_{\{5\},\{3,4,5\}} \\
&Y_{\{1\},\{2\}} = Y_{\{6\},\{1,2,6\}} \\
&\qquad \operatorname{diag}(Y) = e \\
&\qquad Y \succeq 0
\end{aligned}$$

## 4. Theoretical properties of the $R_2$ relaxation

If the given instance of 3-SAT is satisfiable, then it is straightforward to construct a rank-one matrix $Y$ feasible for the $R_2$ relaxation using any satisfying truth assignment. The contrapositive of this statement gives a sufficient condition for proving unsatisfiability using the $R_2$ relaxation.

**Lemma 1.** *Given any instance of 3-SAT, consider the corresponding semidefinite relaxation $R_2$. If the relaxation $R_2$ is infeasible, then the 3-SAT instance is unsatisfiable.*

The rest of this section is concerned with conditions for using $R_2$ to prove satisfiability. More precisely, we prove that for the SDP relaxation $R_2$, if the SDP solver returns a feasible matrix of rank at most two, then the 3-SAT instance is satisfiable. Furthermore, the proof shows how to obtain a satisfying truth assignment from such a feasible matrix. We will make use of the following Lemmas.

**Lemma 2.** *[3, Lemma 3.9] Suppose*

$$\begin{pmatrix} 1 & a & b \\ a & 1 & c \\ b & c & 1 \end{pmatrix} \succeq 0.$$

*Then*

*1. If $a^2 = 1$ then $b = ac$;*
*2. If $b^2 = 1$ then $a = bc$;*
*3. If $c^2 = 1$ then $a = cb$.*

**Lemma 3.** *[3, Lemma 3.11] Suppose that the matrix*

$$\begin{pmatrix} 1 & a & b & c \\ a & 1 & c & b \\ b & c & 1 & a \\ c & b & a & 1 \end{pmatrix}$$

*is positive semidefinite and has rank at most two. Then at least one of a,b,c equals $\pm 1$.*

For every clause $C_j$ with $l(C_j) = 3$, let $Y_{C_j}$ denote the $5 \times 5$ principal submatrix of the matrix variable $Y$ corresponding to the rows and columns of $Y$ indexed by $\{\emptyset, \{i_1\}, \{i_2\}, \{i_3\}, \{i_1, i_2, i_3\}\}$, where $\{i_1, i_2, i_3\} = I_j \cup \bar{I}_j$.

**Lemma 4.** *If $Y$ is feasible for the SDP relaxation and* rank $Y \leq 2$, *then for each clause $C_j$ with $l(C_j) = 3$, the corresponding principal submatrix $Y_{C_j}$ has at least one of the entries corresponding to a pair of variables equal to $\pm 1$.*

*Proof.* Let $i_1$, $i_2$, and $i_3$ denote the three variables in $C_j$, and suppose without loss of generality that $i_1 < i_2 < i_3$. The principal submatrix $Y_{C_j}$ has the form

$$\begin{pmatrix} 1 & Y_{\emptyset,\{i_1\}} & Y_{\emptyset,\{i_2\}} & Y_{\emptyset,\{i_3\}} & Y_{\emptyset,\{i_1,i_2,i_3\}} \\ Y_{\emptyset,\{i_1\}} & 1 & Y_{\{i_1\},\{i_2\}} & Y_{\{i_1\},\{i_3\}} & Y_{\{i_2\},\{i_3\}} \\ Y_{\emptyset,\{i_2\}} & Y_{\{i_1\},\{i_2\}} & 1 & Y_{\{i_2\},\{i_3\}} & Y_{\{i_1\},\{i_3\}} \\ Y_{\emptyset,\{i_3\}} & Y_{\{i_1\},\{i_3\}} & Y_{\{i_2\},\{i_3\}} & 1 & Y_{\{i_1\},\{i_2\}} \\ Y_{\emptyset,\{i_1,i_2,i_3\}} & Y_{\{i_2\},\{i_3\}} & Y_{\{i_1\},\{i_3\}} & Y_{\{i_1\},\{i_2\}} & 1 \end{pmatrix}$$

and since rank $Y \leq 2 \Rightarrow \operatorname{rank} Y_{C_j} \leq 2$, it follows by Lemma 3 that at least one of $Y_{\{i_1\},\{i_2\}}$, $Y_{\{i_1\},\{i_3\}}$, and $Y_{\{i_2\},\{i_3\}}$ equals $\pm 1$. □

By Lemma 4, suppose (without loss of generality) that $Y_{\{i_1\},\{i_2\}} = \delta$, where $\delta^2 = 1$. Then for every row index $I'$ in $\mathcal{O}$ the principal submatrix indexed by $\{I', \{i_3\}, \{i_1, i_2, i_3\}\}$ has the form

$$\begin{pmatrix} 1 & Y_{I',\{i_3\}} & Y_{I',\{i_1,i_2,i_3\}} \\ Y_{I',\{i_3\}} & 1 & \delta \\ Y_{I',\{i_1,i_2,i_3\}} & \delta & 1 \end{pmatrix}$$

and by Lemma 2, it follows that $Y_{I',\{i_3\}} = \delta Y_{I',\{i_1,i_2,i_3\}}$. Hence, the entire column $\{i_1, i_2, i_3\}$ equals $\delta$ times the column $\{i_3\}$. Similarly, the column $\{i_2\}$ equals $\delta$ times the column $\{i_1\}$.

Now let us turn our attention to the constraints enforcing satisfiability. Firstly, observe that $Y(\{i_2\}) = \delta Y(\{i_1\})$ implies that for all the constraints involving terms $Y(I)$ such that $i_2 \in I$, we can replace $Y(I)$ by $\delta Y((I \backslash \{i_2\}) \cup \{i_1\})$ (since column $\{i_2\}$ equals $\delta$ times column $\{i_1\}$). Secondly, consider the clauses $C_j$ of length 3. For each such clause $C_j$, we have the constraint

$$\begin{aligned} &1 - s_{j,i_1} Y(\{i_1\}) - s_{j,i_2} Y(\{i_2\}) - s_{j,i_3} Y(\{i_3\}) + s_{j,i_1} s_{j,i_2} Y(\{i_1, i_2\}) \\ &+ s_{j,i_1} s_{j,i_3} Y(\{i_1, i_3\}) + s_{j,i_2} s_{j,i_3} Y(\{i_2, i_3\}) - s_{j,i_1} s_{j,i_2} s_{j,i_3} Y(\{i_1, i_2, i_3\}) = 0 \end{aligned} \tag{5}$$

(which we have rewritten for convenience). By the previous observations, we have that $Y(\{i_1, i_2\}) = \delta$ implies

$$\begin{aligned} Y(\{i_1, i_2, i_3\}) &= \delta Y(\{i_3\}) \\ Y(\{i_2, i_3\}) &= \delta Y(\{i_1, i_3\}) \\ Y(\{i_2\}) &= \delta Y(\{i_1\}) \end{aligned}$$

Substituting for $Y(\{i_1, i_2, i_3\})$, $Y(\{i_2, i_3\})$, and $Y(\{i_2\})$ in equation (5) yields

$$\begin{aligned} [1 + \delta s_{j,i_1} s_{j,i_2}] - [s_{j,i_1} + \delta s_{j,i_2}] Y(\{i_1\}) - [s_{j,i_3} + \delta s_{j,i_1} s_{j,i_2} s_{j,i_3}] Y(\{i_3\}) \\ + [s_{j,i_1} s_{j,i_3} + \delta s_{j,i_2} s_{j,i_3}] Y(\{i_1, i_3\}) = 0 \end{aligned}$$

which is equivalent to

$$\left[1 + \delta s_{j,i_1} s_{j,i_2}\right] \left[1 - s_{j,i_1} Y(\{i_1\}) - s_{j,i_3} Y(\{i_3\}) + s_{j,i_1} s_{j,i_3} Y(\{i_1, i_3\})\right] = 0. \tag{6}$$

Since $\delta s_{j,i_1} s_{j,i_2} = \pm 1$, we have two cases:

**Case 1.:** $\delta s_{j,i_1} s_{j,i_2} = -1$. Then $1 + \delta s_{j,i_1} s_{j,i_2} = 0$ and therefore the constraint will hold regardless of the values assigned to $Y(\{i_1\})$, $Y(\{i_3\})$, and $Y(\{i_1, i_3\})$. This case occurs when any truth assignment to $x_{i_1}$ and $x_{i_2}$ such that $x_{i_1} x_{i_2} = \delta$ ensures that the clause evaluates to TRUE, regardless of the value of $x_{i_3}$.

**Case 2.:** $\delta s_{j,i_1} s_{j,i_2} = 1$. In this case, we must ensure that the values assigned to $Y(\{i_1\})$, $Y(\{i_3\})$, and $Y(\{i_1, i_3\})$ satisfy the constraint

$$1 - s_{j,i_1} Y(\{i_1\}) - s_{j,i_3} Y(\{i_3\}) + s_{j,i_1} s_{j,i_3} Y(\{i_1, i_3\}) = 0.$$

Finally, we define a reduced SDP as follows: take every constraint arising from Case 2 above, and let the matrix $\tilde{Y}$ be indexed by the set of singletons $\{i_k\}$ such that $Y(\{i_k\})$ appears in at least one of these constraints. The SDP defined by the matrix $\tilde{Y}$ and the constraints arising from Case 2 above, plus the constraint $\tilde{Y} \succeq 0$, is precisely of the form of the Gap relaxation for an instance of 2-SAT. This means that the presence of a $\pm 1$ in each principal submatrix corresponding to a clause of length 3 allows us to reduce the problem to an instance of 2-SAT.

Now recall that we have a matrix $Y$ feasible for the original $R_2$ relaxation, and observe that its principal submatrix indexed by the set of singletons used to define $\tilde{Y}$ is feasible for this reduced SDP. By Theorem 1, this implies that the instance of 2-SAT is satisfiable. Consider a truth assignment that satisfies the 2-SAT instance. For any of the variables in the original instance of 3-SAT that are not present in this truth assignment, either they are to be set equal to $\pm$ one of the variables in this assignment, or they are "free", and can be assigned the value $+1$, say. Finally, construct the rank-one matrix $Y$ corresponding to this truth assignment. Then $Y$ is feasible for the original SDP relaxation, and hence we conclude that the instance of 3-SAT is satisfiable.

Thus, we have proved:

**Theorem 3.** *Given any instance of 3-SAT in CNF, consider the corresponding semidefinite relaxation* $R_2$:

– *If* $R_2$ *is infeasible, then the instance is unsatisfiable.*
– *If* $Y$ *is feasible for* $R_2$ *and* rank $Y \leq 2$, *then the 3-SAT instance is satisfiable.*

## 5. A trio of linearly sized semidefinite relaxations

For any instance of 3-SAT with $n$ variables and $m$ clauses, the SDP relaxation $R_2$ can be viewed as an intermediate relaxation between $R_1$ and $R_3$, and thus it completes a trio of linearly sized SDP relaxations with correspondingly stronger rank guarantees. The characteristics of the relaxations are summarized in Table 1.

The names of the relaxations reflect their increasing strength in the following sense: For $k = 1, 2, 3$, any feasible solution to the relaxation $R_k$ with rank at most $k$ proves satisfiability of the corresponding 3-SAT instance. Furthermore, the increasing values of $k$ also reflect an improving ability to detect unsatisfiability, and an increasing computational time for solving the relaxation. Nonetheless, the dimensions of the relaxations grow only linearly with $n$ and $m$.

**Table 1.** A Trio of Linearly Sized SDP Relaxations for 3-SAT

| Relaxation | Dimension of the matrix variable $Y$ | Number of linear constraints | Sufficient condition on $\text{rank}(Y)$ to deduce SAT |
|---|---|---|---|
| $R_1$ | $n+1$ | $n+m+1$ | $\text{rank}(Y)=1$ |
| $R_2$ | $\leq n+m+1$ | $\leq n+5m+1$ | $\text{rank}(Y)\leq 2$ |
| $R_3$ | $\leq n+4m+1$ | $\leq n+26m+1$ | $\text{rank}(Y)\leq 3$ |

## 6. Computational comparison of the SDP relaxations

We tested the effectiveness of the three relaxations by applying each of them within an enumerative search procedure to prove satisfiability or unsatisfiability of 3-SAT instances. The SDP relaxations were solved using the solver SDPT3 (version 3.0) [34]. Although the SDPs in this paper have all been stated as feasibility problems, in practice we must choose an objective function to be optimized. We used the function $\sum_{i=1}^{n} x_i$ as objective function for all our tests. We did not exploit any bounding information within the enumerative search since we stopped as soon as we found either a feasible solution of sufficiently low rank (and hence a valid truth assignment), or a certificate of infeasibility.

### *6.1. Criteria for Detecting Satisfiability*

In implementing our algorithm, it is important to use the rank conditions to detect that a feasible matrix actually yields a satisfying truth assignment. The rank-one matrices can be detected by applying the following result from [1, Theorem 2.1.1]:

**Theorem 4.** *For any symmetric $n \times n$ positive semidefinite matrix $Y$,*

$$Y_{i,j} = \pm 1 \text{ for all } i,j = 1,\ldots,n \quad \text{if and only if} \quad \text{rank}(Y) = 1.$$

Hence, if the optimal matrix $Y^*$ returned by the solver has all of its entries equal to $\pm 1$, then $Y^*$ directly yields a truth assignment showing that the formula is satisfiable. In our algorithm, the tolerance for determining that an entry of $Y$ would be considered equal to $\pm 1$ was chosen to be 0.999. Thus, if $|Y_{i,j}| \geq 0.999$ then we set $Y_{i,j} = \text{sign}(Y_{i,j})$.

Failing that, we can check for $R_2$ and $R_3$ whether the rank of $Y^*$ is sufficiently small by calculating the eigenvalues of $Y^*$ and checking them against a given tolerance to determine whether they are deemed to be nonzero. (This tolerance was set to $10^{-8}$ in our algorithm.) If the appropriate rank condition holds, then we use the construction in Section 4 to reduce the problem to an instance of 2-SAT. A satisfying truth assignment for this instance of 2-SAT can be obtained from the appropriate principal submatrix of $Y^*$ as described in Section 5 of [12]. Once we have this truth assignment, it is straightforward to extend it to a satisfying truth assignment for the original 3-SAT instance.

### 6.2. Reducing the Dimension of the SDPs

Once one or more variables have been fixed, the SDP relaxation for the resulting instance can be reduced in size by dropping all the clauses which become satisfied by that assignment to the variables. This has a positive impact both on the dimension of the matrix variable and the number of constraints in the SDP. As a consequence, the SDP relaxation quickly becomes more effective as we fix one or more variables during the enumeration process. Obviously, if the current set of fixed variables renders all clauses satisfied or makes any clause unsatisfiable, then no SDP needs to be solved.

### 6.3. Search Strategy

Our search strategy was depth-first search, and we used the following strategy for choosing the branching variable:

1. For the current instance, count the number of occurrences of each variable (or its negation);
2. Consider the variables that attain the highest count, and choose the variable which has smallest absolute value in the solution to the current SDP relaxation;
3. Branch on this variable, and search first the assignment to this variable which immediately satisfies the larger number of clauses.

Any ties in steps 2 and 3 are broken arbitrarily.

The motivation for step 1 is to choose a variable for which the fixing will impact a large number of clauses. Then, in step 2, we choose from among these variables the one with smallest absolute value. The idea behind this heuristic is that for the rank-one matrices corresponding to truth assignments, all the entries equal $\pm 1$; hence entries with small magnitude are less desirable. Finally, step 3 is a greedy choice that uses the fixing of the variable to satisfy as many clauses as possible immediately.

### 6.4. Summary of Computational Results

The algorithm was implemented in Matlab on a 2.4 GHz Pentium IV with 1.5Gb. We solved both satisfiable and unsatisfiable instances of 3-SAT from the Uniform Random-3-SAT benchmarks available at [21]. We considered two sets of problems having $n = 50$ and $n = 75$ variables and $m = 218$ and $m = 325$ clauses respectively. We ran a total of 40 instances of 3-SAT from each set, half of them satisfiable and the other half unsatisfiable. For all the relaxations, the algorithm was stopped after 2 hours on the instances with 50 variables, and after 3 hours on the instances with 75 variables. The computational results averaged over the 20 instances of each type are reported in Tables 2 and 3. These results are for small problems, but they clearly illustrate the tradeoffs involved in the choice of SDP relaxation as well as the advantage of the $R_2$ relaxation over the other two relaxations when applied within an enumerative algorithm. We make the following observations:

– The Gap relaxation is solved most quickly of all three, but we see that the enumerative algorithm using it has to reach a large depth in the search tree

**Table 2.** Computational results for satisfiable 3-SAT instances

Instances with $n = 50$ variables and $m = 218$ clauses

| SDP used | Statistics of initial SDP averaged over 20 instances | | | # of instances solved (2-hour timeout) | Performance statistics averaged over the solved instances | | | |
|---|---|---|---|---|---|---|---|---|
| | # linear constraints | Dimension of $Y$ | CPU seconds to solve | | # calls to SDP solver | Deepest node reached | Depth of valid truth assignment found | CPU seconds for proof of SAT |
| Gap | 269 | 51 | 2 | 20 | 664 | 42 | 41 | 603 |
| $R_2$ | 486 | 268 | 28 | 20 | 28 | 12 | 12 | 199 |
| $R_3$ | 5104 | 775 | 1205 | 11 | 11 | 8 | 8 | 5165 |

Instances with $n = 75$ variables and $m = 325$ clauses

| SDP used | Statistics of initial SDP averaged over 20 instances | | | # of instances solved (3-hour timeout) | Performance statistics averaged over the solved instances | | | |
|---|---|---|---|---|---|---|---|---|
| | # linear constraints | Dimension of $Y$ | CPU seconds to solve | | # calls to SDP solver | Deepest node reached | Depth of valid truth assignment found | CPU seconds for proof of SAT |
| Gap | 401 | 76 | 3 | 11 | 2498 | 62 | 59 | 4083 |
| $R_2$ | 725 | 400 | 29 | 20 | 91 | 18 | 17 | 1259 |
| $R_3$ | 7883 | 1220 | 4604 | 0 | – | – | – | – |

before it stops. As a result, its total time is higher than that of the algorithm using $R_2$.

– The opposite happens with the algorithm using $R_3$: each SDP takes much longer to solve, but the depth reached in the search tree is lower than for the other two relaxations.

We conclude that, in comparison with the other two relaxations, the $R_2$ relaxation is the most effective, and it can be routinely used to prove both satisfiability and unsatisfiability for instances with a few hundred clauses. However, we must mention that the computational time for the algorithm using $R_2$ still increases quite rapidly. To illustrate this, we applied this algorithm to 40 instances of 3-SAT with $n = 100$ and $m = 430$ from the same set of benchmarks, evenly divided between satisfiable and unsatisfiable instances. The algorithm was allowed to run to completion on all instances. The computational results averaged over each set of 20 instances are reported in Table 4. Notice that proofs of satisfiability require over one hour on average, and proofs of unsatisfiability over six hours. Indeed, the computational time for any SDP-based algorithm is dominated by the effort required to solve the SDPs. It is important to note that we used a general-purpose solver which does not take advantage of the structure of the SDPs. Although important advances have been made in the development of algorithms for solving SDPs by taking advantage of structure [7,14,31,20,23,24,9,36], none of the techniques proposed so far is of help in solving the SAT relaxations. Current

**Table 3.** Computational results for unsatisfiable 3-SAT instances

Instances with $n = 50$ variables and $m = 218$ clauses

| SDP used | Statistics of initial SDP averaged over 20 instances | | | # of instances solved (2-hour timeout) | Performance statistics averaged over the solved instances | | |
|---|---|---|---|---|---|---|---|
| | # linear constraints | Dimension of $Y$ | CPU seconds to solve | | # calls to SDP solver | Deepest node reached | CPU seconds for proof of UnSAT |
| Gap | 269 | 51 | 2 | 20 | 363 | 23 | 526 |
| $R_2$ | 485 | 267 | 12 | 20 | 41 | 6 | 337 |
| $R_3$ | 5090 | 773 | 1189 | 2 | 8 | 3 | 6139 |

Instances with $n = 75$ variables and $m = 325$ clauses

| SDP used | Statistics of initial SDP averaged over 20 instances | | | # of instances solved (3-hour timeout) | Performance statistics averaged over the solved instances | | |
|---|---|---|---|---|---|---|---|
| | # linear constraints | Dimension of $Y$ | CPU seconds to solve | | # calls to SDP solver | Deepest node reached | CPU seconds for proof of UnSAT |
| Gap | 401 | 76 | 3 | 6 | 2626 | 41 | 7564 |
| $R_2$ | 725 | 400 | 30 | 20 | 194 | 11 | 2955 |
| $R_3$ | 7886 | 1220 | 4628 | 0 | - | - | - |

research is considering how the structure of the $R_2$ and $R_3$ relaxations could be specifically exploited within an SDP solver.

## 7. Future research directions

In this paper, we have proposed a new SDP relaxation for SAT and presented computational results showing that it is more effective than previous relaxations in the literature when considering an SDP-based enumerative approach for solving instances of SAT in conjunctive normal form. Based on the computational results in [2], it is likely that an SDP-based approach has the potential to complement existing SAT algorithms, and this new relaxation is another step in our ongoing research towards a practical SDP-based algorithm for SAT.

From this perspective, we have two particularly important directions for future research. Firstly, the enumerative algorithm could be significantly improved by the application of more sophisticated techniques known in the SAT area, such as splitting and resolution (see e.g. [17, Section 6] and the references therein). Secondly, the computational time for the SDP-based algorithms is dominated by the effort required to solve the SDPs. We have solved all the SDPs using a general-purpose solver which does not take advantage of the structure of the SDP relaxations for SAT. Future research will consider how the structure of the $R_2$ and $R_3$ relaxations could be specifically exploited.

**Table 4.** Computational results for the $R_2$-based algorithm on larger instances

Satisfiable instances with $n = 100$ variables and $m = 430$ clauses

| SDP used | Statistics of initial SDP averaged over 20 instances | | | Performance statistics averaged over the solved instances | | | |
|---|---|---|---|---|---|---|---|
| | # linear constraints | Dimension of $Y$ | CPU seconds to solve | # calls to SDP solver | Deepest node reached | Depth of valid truth assignment found | CPU seconds for proof of SAT |
| $R_2$ | 961 | 531 | 61 | 348 | 35 | 33 | 5053 |

Unsatisfiable instances with $n = 100$ variables and $m = 430$ clauses

| SDP used | Statistics of initial SDP averaged over 20 instances | | | Performance statistics averaged over the solved instances | | |
|---|---|---|---|---|---|---|
| | # linear constraints | Dimension of $Y$ | CPU seconds to solve | # calls to SDP solver | Deepest node reached | CPU seconds for proof of SAT |
| $R_2$ | 961 | 531 | 60 | 906 | 15 | 24924 |

**Acknowledgements.** The author thanks Franz Rendl, Monique Laurent, and Etienne de Klerk for several helpful discussions.

## References

1. Anjos MF (2001) New Convex Relaxations for the Maximum Cut and VLSI Layout Problems. PhD thesis, University of Waterloo. Published online at http://etd.uwaterloo.ca/etd/manjos2001.pdf
2. Anjos MF An improved semidefinite programming relaxation for the satisfiability problem. Math. Program., (Ser. A), to appear
3. Anjos MF, Wolkowicz H (2002) Geometry of semidefinite max-cut relaxations via matrix ranks. J. Comb. Optim., 6(3):237–270
4. Anjos MF, Wolkowicz H (2002) Strengthened semidefinite relaxations via a second lifting for the max-cut problem. Discrete Appl. Math., 119(1–2):79–106
5. Apswall B, Plass MF, Tarjan RE (1979) A linear-time algorithm for testing the truth of certain quantified boolean formulas. Inform. Process. Lett., 8(3):121–123
6. Balas E, Ceria S, Cornuéjols G (1993) A lift-and-project cutting plane algorithm for mixed 0–1 programs. Math. Program., 58(3, Ser. A):295–324
7. Benson SJ, Ye Y, Zhang X (2000) Solving large-scale sparse semidefinite programs for combinatorial optimization. SIAM J. Optim., 10(2):443–461 (electronic)
8. Bienstock D, Zuckerberg M (2002) Subset algebra lift operators for 0-1 integer programming. Technical Report CORC 2002-01, Columbia University
9. Burer S (2002) Semidefinite programming in the space of partial positive semidefinite matrices. Technical report, Department of Management Sciences, University of Iowa,
10. Conforti M, Cornuéjols G (1995) A class of logic problems solvable by linear programming. J. ACM, 42(5):1107–1113
11. de Klerk E, van Maaren H (2003) On semidefinite programming relaxations of $(2 + p)$-SAT. Ann. Math. Artif. Intell., 37(3):285–305
12. de Klerk E, van Maaren H, Warners JP (2000) Relaxations of the satisfiability problem using semidefinite programming. J. Automat. Reason., 24(1–2):37–65

13. Deza MM, Laurent M (1997) Geometry of cuts and metrics, volume 15 of Algorithms and Combinatorics. Springer-Verlag, Berlin
14. Fukuda M, Kojima M, Murota K, Nakata K (2000/01) Exploiting sparsity in semidefinite programming via matrix completion I: General framework. SIAM J. Optim., 11(3):647–674 (electronic)
15. Goemans MX, Williamson DP (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. Assoc. Comput. Mach., 42(6):1115–1145
16. Grigoriev D, Hirsch EA, Pasechnik DV (2002) Complexity of semialgebraic proofs. Moscow Math. J., 2(4):647–679
17. Gu J, Purdom PW, Franco J, Wah BW (1997) Algorithms for the satisfiability (SAT) problem: a survey. In Satisfiability problem: theory and applications (Piscataway, NJ, 1996), pages 19–151. Amer. Math. Soc., Providence, RI
18. Halperin E, Zwick U (2001) Approximation algorithms for MAX 4-SAT and rounding procedures for semidefinite programs. J. Algorithms, 40(2):184–211
19. Helmberg C `http://www-user.tu-chemnitz.de/~helmberg/semidef.html`.
20. Helmberg C (October 2001) A cutting plane algorithm for large scale semidefinite relaxations. Technical Report ZR-01-26, Konrad-Zuse-Zentrum Berlin. To appear in the Padberg Festschrift "The Sharpest Cut", SIAM
21. Hoos HH, Stützle T (2000) Satlib: An online resource for research on sat. In SAT 2000, pages 283–292. IOS Press, SATLIB is available online at www.satlib.org.
22. Karloff H, Zwick U (1997) A 7/8-approximation algorithm for MAX 3SAT? In Proc. of 38th FOCS, pages 406–415
23. Kočvara M, Stingl M (2001) PENNON - a generalized augmented lagrangian method for semidefinite programming. Technical Report 286, Institute of Applied Mathematics, University of Erlangen
24. Kočvara M, Stingl M (2002) PENNON - a code for convex nonlinear and semidefinite programming. Technical Report 290, Institute of Applied Mathematics, University of Erlangen
25. Lasserre JB (2000) Optimality conditions and LMI relaxations for 0–1 programs. Technical report, LAAS-CNRS, Toulouse, France
26. Lasserre JB (2002) An explicit equivalent positive semidefinite program for nonlinear 0–1 programs. SIAM J. Optim., 12(3):756–769 (electronic)
27. Laurent M Semidefinite relaxations for max-cut. In M. Grötschel, editor, The Sharpest Cut, Festschrift in Honor of M. Padberg's 60th Birthday. SIAM, to appear.
28. Laurent M, Rendl F Semidefinite programming and integer programming. In G. Nemhauser K. Aardal and R. Weismantel, editors, Handbook on discrete optimization. Elsevier, to appear
29. Lovász L (1979) On the Shannon capacity of a graph. IEEE Trans. Inform. Theory, 25(1):1–7
30. Lovász L, Schrijver A (1991) Cones of matrices and set-functions and 0-1 optimization. SIAM J. Optim., 1(2):166–190
31. Nakata K, Fujisawa K, Fukuda M, Kojima M, Murota K (2003) Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results. Math. Program., 95:303–327
32. Parrilo PA (2003) Semidefinite programming relaxations for semialgebraic problems. Math. Programming, 96(2, Ser. B):293–320
33. Sherali HD, Adams WP (1990) A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J. Discrete Math., 3(3):411–430
34. Toh KC, Todd MJ, Tütüncü RH (1999) SDPT3—a MATLAB software package for semidefinite programming, version 1.3. Optim. Methods Softw., 11/12(1–4):545–581
35. van Maaren H (1999) Elliptic approximations of propositional formulae. Discrete Appl. Math., 96/97:223–244
36. Wolkowicz H Solving semidefinite programs using preconditioned conjugate gradients. Optim. Methods Softw., to appear
37. Wolkowicz H, Saigal R, Vandenberghe L, editors. (2000) Handbook of semidefinite programming. Kluwer Academic Publishers, Boston, MA
38. Zwick U (1999) Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems. In Proceedings of 31st STOC, pages 679–687