On solving the MAX-SAT using sum of squares

Lennart Sinjorgo * Renata Sotirov †

Abstract

We consider semidefinite programming (SDP) approaches for solving the maximum satisfiability problem (MAX-SAT) and the weighted partial MAX-SAT. It is widely known that SDP is well-suited to approximate the (MAX-)2-SAT. Our work shows the potential of SDP also for other satisfiability problems, by being competitive with some of the best solvers in the yearly MAX-SAT competition. Our solver combines sum of squares (SOS) based SDP bounds and an efficient parser within a branch & bound scheme.

On the theoretical side, we propose a family of semidefinite feasibility problems, and show that a member of this family provides the rank two guarantee. We also provide a parametric family of semidefinite relaxations for the MAX-SAT, and derive several properties of monomial bases used in the SOS approach. We connect two well-known SDP approaches for the (MAX)-SAT, in an elegant way. Moreover, we relate our SOS-SDP relaxations for the partial MAX-SAT to the known SAT relaxations.

Keywords SAT, MAX-SAT, weighted partial MAX-SAT, semidefinite programming, sum of squares, Peaceman-Rachford splitting method

AMS subject classifications. 90C09, 90C22, 90C23.

1 Introduction

In this paper, we investigate semidefinite programming (SDP) approaches for the satisfiability problem, (SAT), maximum satisfiability problem (MAX-SAT) and their variants. Given a logical proposition, built from a conjunction of clauses, the SAT asks whether there exists a truth assignment to the variables such that all clauses are satisfied. The optimization variant of SAT, known as the MAX-SAT, is to determine a truth assignment which satisfies the largest number of clauses.

The SAT is a central problem in mathematical logic and computer science and finds various applications, including software or hardware verification [36] and planning in artificial intelligence [27]. Cook [12] proved that SAT is $\mathcal{NP}$-complete in 1971. The SAT was the first problem proven to be $\mathcal{NP}$-complete, which implies that any problem contained in the complexity class $\mathcal{NP}$ can be efficiently recast as a SAT instance. Thus, algorithms for the SAT can also solve a wide variety of other problems, such as timetabling [9, 17] and product line engineering [37].

SDP approaches to the SAT are proposed first by de Klerk et al. [13], and later extended by Anjos [3]–[7]. Goemans and Williamson [18] were first to apply SDP to the MAX-SAT. They showed that for a specific class of MAX-SAT instances, known as MAX-2-SAT (in the MAX-k-SAT, each clause is a disjunction of at most k variables), the MAX-SAT is equivalent to optimizing a multivariate quadratic polynomial, which is naturally well suited for semidefinite relaxations. In the same paper, Goemans and Williamson proposed a 0.878-approximation algorithm for the MAX-2-SAT based on SDP. This result was later improved to 0.940 in [33]. Further, Karloff and Zwick [26] obtained an optimal 7/8 approximation algorithm for the MAX-3-SAT, and Halperin and Zwick [20] a nearly optimal approximation algorithm for the MAX-4-SAT. In [51], van Maaren et al. exploit sum of squares (SOS) optimization to compute bounds for the MAX-SAT.

Despite the great success in designing approximation algorithms using SDP, most modern MAX-SAT solvers do not exploit SDP. A possible reason for this is the fact that medium to large size SDP problems are computationally challenging to solve. Interior point methods, the conventional approach for solving SDPs, struggle from large memory requirements and prohibitive computation time per iteration already for medium size SDPs. Recently, first order methods such as the alternating direction method of

*Tilburg University, Department of Econometrics and OR, CentER, The Netherlands, l.m.sinjorgo@tilburguniversity.edu

†Tilburg University, Department of Econometrics and OR, CentER, The Netherlands, r.sotirov@tilburguniversity.edu

multipliers (ADMM) [11, 16] and the Peaceman-Rachford splitting method (PRSM) [42] showed a great success in solving SDPs, see e.g., [14, 19, 39]. Motivated by those results, we design a MAX-SAT solver that incorporates SDP bounds and the PRSM within a branch & bound (B&B) scheme.

In particular, we further exploit the SOS approach from [51] to derive SOS-based SDP relaxations that provide strong upper bounds to the optimal MAX-SAT solution. The derived SDP relaxations are strengthened SDP duals of the Goemans and Williamson MAX-SAT relaxation. The strength of the upper bounds and the required time to compute the relaxations depend on the chosen monomial basis. We experiment with different monomial bases and propose a class of bases that provide good trade-offs between these effects. Moreover, we derive several properties of monomial bases that are exploited in the design of our solver. We extend the SOS approach to the weighted partial MAX-SAT, a variant of the MAX-SAT in which clauses are divided in soft and hard clauses. Here, the goal is to maximize the weighted sum of soft clauses, while satisfying all the hard clauses. We strengthen SDP bounds for the weighted partial MAX-SAT using the SAT resolution rule. To the best of our knowledge, we are the first to exploit SDP for solving the weighted partial MAX-SAT.

We show that the Peaceman-Rachford splitting method is well suited for exploiting the structure of the SOS-based SDP relaxations. Therefore, we implement the PRSM to (approximately) solve large-scale SDP relaxations and obtain upper bounds for the (weighted partial) MAX-SAT. The resulting algorithm is very efficient, e.g., it can compute upper bounds with matrix variables of order 1800 in less than 2 minutes, and for matrices of order 2400 in less than 4 minutes. Our numerical results show that the upper bounds are strong, in particular when larger monomial bases is used. We also exploit the output of the PRSM to efficiently compute lower bounds for the MAX-SAT.

We design an SOS-SDP based MAX-SAT solver (named SOS-MS) that exploits SOS-based SDP relaxations and the PRSM. SOS-MS is one of the first SDP-based MAX-SAT solvers. The only alternative SDP-based solver is the MIXSAT algorithm [52] that is designed to solve MAX-2-SAT instances. SOS-MS is able to solve (weighted partial) MAX-k-SAT instances, for $k \leq 3$. To solve a MAX-SAT instance, SOS-MS has to approximately solve multiple SDP subproblems. A crucial component of SOS-MS is therefore its ability to quickly construct the programme parameters of the required SDPs, i.e., the process of *parsing*. We design an efficient parsing method, which is also applicable to other problems and publicly available. Another efficient feature of our solver is warm starts. Namely, our solver uses the approximate PRSM solution at a node, as warm starts for the corresponding children's node. We are able to solve a variety of MAX-SAT instances in a reasonable time, while solving some instances faster than the best solvers in the Eleventh Evaluation of Max-SAT Solvers (MSE-2016). Moreover, we solve three previously unsolved MAX-3-SAT instances from the MSE-2016. We are first to use SDP for solving weighted partial MAX-SAT. Our results provide new perspectives on solving the MAX-SAT, and all its variants, by using SDPs.

Our paper provides various theoretical results. We propose a family of semidefinite feasibility problems, and show that one member of this family provides the rank two guarantee. That is, whenever the semidefinite relaxation admits a feasible matrix of rank two or less, the underlying SAT instance is satisfiable. This result relates to a similar rank two guarantee result by Anjos [4]. The rank value can be seen as a measure of the strength of the relaxation. We also provide a parametric family of semidefinite relaxations for the (weighted partial) MAX-SAT. This parameter can be finely tuned to determine the strength of the relaxation, and any such relaxation can easily be incorporated within SOS-MS. This allows the solver to be adapted per (class of) problem instances.

Further, we show how the SOS approach to the MAX-SAT of van Maaren et al. [51] and here generalized moment relaxations of the SAT due to Anjos [3]-[6] are related. This is done by exploiting the duality theory of the moment and SOS approaches. Our result generalizes a result by van Maaren et al. [51], who showed the connection between the two approaches only for restricted cases. By exploiting duality theory, we also relate the SOS relaxations for the partial MAX-SAT problem to the SAT relaxations from [4].

Lastly, we investigate MAX-SAT resolution, a powerful technique used by many MAX-SAT solvers [1], in relation to the SDP approach to the MAX-SAT. Standard MAX-SAT solvers use resolution to determine upper bounds on the MAX-SAT solution, while SOS-MS determines upper bounds through SDP. We show how resolution is related to the monomial basis. We also show how the SAT resolution can be exploited for the weighted partial MAX-SAT.

This paper is organized as follows. We provide notation and preliminaries in Section 1.1 and assumptions in Section 1.2. Section 2 provides an overview of the Goemans and Williamson approach [18]

to the MAX-SAT. Section 3 first outlines previous SDP approaches to the SAT, and then generalizes them. Section 4 provides the details of the SOS theory, applied to the MAX-SAT. We also derive various properties of monomial bases in that section. Section 5 concerns the combination of MAX-SAT resolution and SOS. In Section 6, we show how two SDP approaches to (MAX-)SAT, i.e., [4] and [51], are connected. Section 7 introduces the PRSM for SOS. We extend the SOS approach to the weighted partial MAX-SAT and connect the resulting programme to the relaxations in [4], in Section 8. Section 9 provides an overview and pseudocode of our solver SOS-MS. Section 10 presents numerical results that include SOS-SDP bounds and performance of SOS-MS. Concluding remarks are given in Section 11.

1.1 Preliminaries and notation

For any $n \in \mathbb{N}$, we write $[n] = \{1, \ldots, n\}$. We denote by ϕ a propositional formula, in variables x_1 up to x_n, and assume that ϕ is in *conjunctive normal form* (CNF). That is, ϕ is given by a conjunction of m clauses,

$$\phi = \bigwedge_{j=1}^{m} C_j. \tag{1}$$

We will mostly use n to refer to the number of variables and m to refer to the number of clauses. Each clause C_j is a disjunction of (possibly negated) variables. We define each clause C_j as a subset of $[n]$, indicating the variables appearing in C_j. Moreover, we define $I_j^+ \subseteq C_j$ as the set of unnegated variables appearing in C_j. Similarly, $I_j^- \subseteq C_j$ is defined as the set of negated variables appearing in C_j. Thus, the clause associated with C_j reads

$$\bigvee_{i \in I_j^+} x_i \vee \bigvee_{i \in I_j^-} \neg x_i. \tag{2}$$

We refer to both x_i and $\neg x_i$ as literals. For example, the literal $\neg x_i$ is true if x_i is `false`. We denote the length of a clause by ℓ_j, thus $\ell_j := |C_j|$. We say that ϕ constitutes a (MAX-)k-SAT instance if $\max_{j \in [m]} \ell_j = k$.

The SAT is to decide, given ϕ, whether a satisfying truth assignment to the variables x_i, $i \in [n]$ exists. The MAX-SAT is to find an assignment which satisfies the largest number of clauses.

We associate to each clause a vector $a_j \in \{0, \pm 1\}^n$, having entries $a_{j,i}$ according to

$$a_{j,i} = \begin{cases} -1, & \text{if } i \in I_j^-, \\ 0, & \text{if } i \notin I_j^+ \cup I_j^-, \\ 1, & \text{if } i \in I_j^+. \end{cases} \tag{3}$$

We write $\mathcal{S}^n$ for the set of symmetric $n \times n$ matrices and $\mathcal{S}_+^n$ for the cone of symmetric positive semidefinite matrices of size $n \times n$. If the context is clear, the superscript n will be omitted. We denote by $\mathbf{1}_n \in \mathbb{R}^n$ and $\mathbf{0}_n \in \mathbb{R}^n$, vectors of all ones and zeroes, respectively (subscripts are omitted when the context is clear). The identity matrix $I_n \in \mathcal{S}^n$ has as columns the unit vectors e_i, $i \in [n]$. Matrix $\mathbf{0}_{m \times n}$ denotes the zero matrix of m rows and n columns.

For $X, Y \in \mathcal{S}$, we define the trace inner product as $\langle X, Y \rangle := \mathrm{Tr}(XY)$. We write $X \succeq Y$ if and only if $X - Y \in \mathcal{S}_+$. The Frobenius norm of a symmetric matrix X is given by

$$\|X\|_F = \sqrt{\langle X, X \rangle}. \tag{4}$$

For $X \in \mathcal{S}$ and $\mathcal{M} \subseteq \mathcal{S}$, we set

$$\mathcal{P}_{\mathcal{M}}(X) := \operatorname*{argmin}_{Z \in \mathcal{M}} \|Z - X\|_F, \tag{5}$$

as the projection of X onto $\mathcal{M}$.

For $X \in \mathbb{R}^{n \times n}$, $\mathrm{diag}(X) \in \mathbb{R}^n$ denotes the vector equal to the diagonal of X. For $x \in \mathbb{R}^n$, we write $\|x\| := \sqrt{x^\top x}$ and

$$x^\alpha := \prod_{i \in \alpha} x_i, \tag{6}$$

for some $\alpha \subseteq [n]$. We evaluate the empty product as 1. Note that (6) differs from the more common notation in SOS literature, where the α are considered as vectors in $\mathbb{N}^n$, see e.g., [31].

1.2 Assumptions

In the rest of this work, we assume that all logical propositions ϕ, on n variables and m clauses, satisfy the following three properties:

1. $I_j^+ \cap I_j^- = \emptyset$, $\forall j \in [m]$,
2. $|C_j| \geq 2$, $\forall j \in [m]$,
3. Each variable is contained it at least 2 clauses,

along with ϕ being in CNF. We explain now that properties 1 and 3 can be assumed without loss of generality. Property 1 states that a clause cannot contain both the unnegated and negated variants of a variable. If that were not the case, the clause is trivially satisfied and can be removed from ϕ without loss of generality. For property 3, if we have that a variable occurs in exactly one clause, say C_j, we can set that variable to the truth value such that C_j is satisfied and remove C_j from ϕ.

Property 2 can be assumed for SAT instances. If a SAT instance contains a clause C_j with $|C_j| = 1$ (such a clause is known as a *unit clause*), the literal in C_j must be satisfied. The variable corresponding to this literal can thus be given an appropriate truth value and ϕ can be reduced (such a reduction of ϕ is referred to as *unit resolution*). For MAX-SAT instances, it is possible that an optimal truth assignment might leave unit clauses unsatisfied. We note however, that the MAX-SAT benchmark instances we consider satisfy all the above assumptions.

2 The MAX-SAT formulation and relaxation

We outline the approach of Goemans-Williamson for formulating the MAX-SAT as a polynomial optimization problem. We also present their SDP relaxation for the MAX-2-SAT.

Let $x_1, x_2, \ldots, x_n$ be the variables of the MAX-SAT instance, given by a logical proposition ϕ. We thus assume that ϕ is given in conjunctive normal form, see (1), and contains m clauses. As customary in the SDP SAT literature, we associate $+1$ with `true` and -1 with `false`. An assignment of the x_i values in $\{\pm 1\}$ is referred to as a truth assignment. As proposed by Goemans and Williamson [18], we define a truth function $v : \{\pm 1\}^n \to \{0, 1\}$, such that, given a logical proposition ϕ', evaluated for some truth assignment, $v(\phi') = 1$ if and only if ϕ' is satisfied, and 0 otherwise. This property uniquely determines v, i.e.,

$$v(x_i) = \frac{1 + x_i}{2} \text{ and } v(\neg x_i) = \frac{1 - x_i}{2}.$$

And in general, for a clause $C_j \subseteq [n]$ of length ℓ_j, we have

$$v(C_j) := 1 - \prod_{i \in I_j^+} v(\neg x_i) \prod_{i \in I_j^-} v(x_j) = 1 - 2^{-\ell_j} \Big(\sum_{\gamma \subseteq C_j} (-1)^{|\gamma|} a_j^\gamma x^\gamma \Big), \tag{7}$$

for a_j as in (3), and both a_j^γ and x^γ as in (6). The last equality in (7) follows from the product expansion of $v(C_j)$, as shown in Proposition 1 in [6]. In [18], an extra variable $x_0 \in \{\pm 1\}$ is defined, with the purpose of deciding the truth value, that is, ϕ' is true if and only if $v(\phi') = x_0$. We set $x_0 = 1$ without loss of generality for sake of clarity. The MAX-SAT, given by ϕ, is to maximize the following polynomial:

$$v_\phi := \sum_{j \in [m]} v(C_j) = \sum_{\alpha \subseteq [n]} v_\phi^\alpha x^\alpha, \tag{8}$$

subject to $x_i \in \{\pm 1\}$ for all i, and for appropriate $v_\phi^\alpha \in \mathbb{R}$, and x^α as in (6). Observe that v_ϕ is a kth degree polynomial if ϕ represents a MAX-k-SAT instance. A MAX-2-SAT instance thus corresponds to a quadratic polynomial, and is therefore well suited for approximated by SDP. We return to v_ϕ in Section 6.

Assuming now that ϕ represents a MAX-2-SAT instance on n variables, the corresponding MAX-2-SAT can be formulated as

$$\begin{aligned} \max \quad & \langle W, X \rangle \\ \text{s.t.} \quad & \operatorname{diag}(X) = \mathbf{1}, \\ & X \succeq 0,\ X \in \{\pm 1\}^{(n+1)\times(n+1)}, \end{aligned} \tag{9}$$

where $\langle W, X\rangle$ describes the quadratic polynomial v_ϕ. Observe that the constraints of (9) enforce X to satisfy $X = xx^\top$, for some $x \in \{\pm 1\}^{n+1}$. The size of this vector x is one more than the number of variables n, to account for the additional truth value variable x_0.

A semidefinite relaxation of (9) is obtained by omitting the integrality constraint, or equivalently non-convex rank one constraint. This constitutes the well-known Goemans-Williamson [18] SDP relaxation of the MAX-2-SAT. That is,

$$\begin{array}{ll} \max & \langle W, X\rangle \\ \text{s.t.} & \operatorname{diag}(X) = \mathbf{1}, \\ & X \in \mathcal{S}_+^{n+1}. \end{array} \tag{10}$$

Goemans and Williamson showed that the optimal matrix for (10) can be used to obtain a 0.878-approximation algorithm to the MAX-2-SAT. Assuming $\mathcal{P} \neq \mathcal{NP}$, for any $\varepsilon > 0$ there exists no $(\frac{21}{22} + \varepsilon)$ approximation algorithm to the MAX-2-SAT [21]. Karloff and Zwick [26] introduce a canonical way of obtaining SDP relaxations for any MAX-SAT, that is exploited to obtain approximation algorithms to the MAX-3-SAT and the MAX-4-SAT in [26] and [20], respectively. To solve MAX-2-SAT instances, rather than approximate, Wang and Zico Kolter [52] propose the MIXSAT algorithm, which combines (10) with a B&B scheme.

3 The SAT as semidefinite feasibility problem

In this section we first present a brief overview of the work done by de Klerk et al. [13] and Anjos [3, 4, 5, 6, 7]. Their relaxations of the SAT involve semidefinite feasibility problems: infeasibility of the semidefinite programme implies unsatisfiability of the corresponding SAT instance. The difference between relaxations is the size of the SDP variable, and the method of encoding the structure of the SAT instance in the SDP relaxation. Then, we propose a family of semidefinite feasibility problems, that contains relaxations from [3]–[7] and [13] as special cases, and show that a particular member of the family provides a rank-two guarantee, see Theorem 1.

We reconsider first programme (10), which attempts to satisfy the maximum number of clauses through its objective function. For the SAT specifically, one might move the clause satisfaction part from the objective to the feasible set of a semidefinite programme. This idea was first proposed by de Klerk et al. [13] in 2000, and was later extended by Anjos [3]. To be precise: de Klerk et al. propose the so called GAP relaxation, or GAP for short, which is a semidefinite feasibility problem, given by

$$\begin{array}{ll} \text{find} & Y \in \mathcal{S}_+^n,\ y \in \mathbb{R}^n \\ \text{s.t.} & a_j^\top Y a_j - 2a_j^\top y \leq \ell_j(\ell_j - 2),\ \forall j \in [m], \\ & \operatorname{diag}(Y) = \mathbf{1}, \\ & Y \succeq yy^\top, \end{array} \tag{GAP}$$

for a_j as in (3). It is noted in [13], that for $\ell_j \leq 2$, the corresponding inequalities in GAP relaxation may be changed to equalities. The GAP relaxation is suited for instances that contain a clause of length two. If $\ell_j \geq 3$, $\forall j \in [m]$, then $(Y, y) = (I, \mathbf{0})$ is always feasible for GAP, whether the underlying SAT instance is satisfiable or not.

We now state the SDP relaxations of the SAT by Anjos [3]-[7] that are not restricted to the lengths of the clauses in instances. Let ϕ be a proposition on n variables and m clauses and $x \in \{\pm 1\}$ the truth assignment to the variables. Consider a family of subsets $\mathcal{F} = \{\alpha_1, \ldots, \alpha_s\}$, let $\mathbf{x} = (x^{\alpha_1}, \ldots, x^{\alpha_s})^\top$, and define $Y := \mathbf{x}\mathbf{x}^\top$. It is clear that $\operatorname{rank}(Y) = 1$, $\operatorname{diag}(Y) = \mathbf{1}$, and $Y \succeq 0$. Later, to obtain a semidefinite relaxation of the SAT, we omit the rank one constraint.

We index the matrix Y with the elements of $\mathcal{F}$, and define for all subsets γ contained in some clause of ϕ, the expression

$$Y(\gamma) := Y_{\alpha,\beta},\ \text{for some } \alpha,\ \beta \in \mathcal{F} \text{ jointly contained in a single clause, such that } \alpha \triangle \beta = \gamma, \tag{11}$$

where $\triangle$ is the symmetric difference operator, which is induced by the fact that, for $x \in \{\pm 1\}^n$, we have $Y_{\alpha,\beta} = x^\alpha x^\beta = x^{\alpha \triangle \beta} = x^\gamma$, see (6). In general, $Y(\emptyset)$ refers to a diagonal entry of Y, hence, $Y(\emptyset) = 1$. We may have $Y(\gamma) = Y_{\emptyset,\gamma}$, and we assume that, for all γ contained in a clause of ϕ, we can always find α and β as in (11).

The expression $Y(\gamma)$ can refer to multiple entries of Y. By construction of Y, these entries are equal. Stated formally, we have $Y \in \cap_{j\in[m]}\Delta_j$, where

$$\Delta_j := \{Y \in \mathcal{S} \,|\, Y_{\alpha_1,\beta_1} = Y_{\alpha_2,\beta_2} \ \forall\, (\alpha_1,\alpha_2,\beta_1,\beta_2) \in \mathcal{F} \text{ such that } \alpha_1\triangle\beta_1 = \alpha_2\triangle\beta_2 \subseteq C_j\}. \tag{12}$$

Observe that the sets Δ_j do not capture all equalities present in Y, due to the restriction $\alpha_1\triangle\beta_1 = \alpha_2\triangle\beta_2 \subseteq C_j$. In this section, we choose to include only the equalities captured by Δ_j. This keeps our work in line with previous relaxations by Anjos [4], and these equalities suffice to prove the main theorem in this section, see Theorem 1. In Section 6, we consider an SDP relaxation of the SAT which considers all equalities present in Y.

If x is a satisfying assignment to ϕ, then $v(C_j) = 1$, see (7), for all $j \in [m]$. We can rewrite this constraint in terms of $Y(\gamma)$, see (11). We now omit the rank one constraint on Y, to obtain the following semidefinite feasibility programme, denoted $R_{\mathcal{F}}(\phi)$:

$$\begin{aligned} \text{find} \quad & Y \in \mathcal{S}_+^{|\mathcal{F}|} \\ \text{s.t.} \quad & \sum_{\gamma\subseteq C_j} (-1)^{|\gamma|}\, a_j^{\gamma}\, Y(\gamma) = 0, \ \text{ for all } j \in [m], \\ & \operatorname{diag}(Y) = \mathbf{1},\, Y \in \bigcap_{j\in[m]} \Delta_j. \end{aligned} \tag{$R_{\mathcal{F}}(\phi)$}$$

The programme $R_{\mathcal{F}}(\phi)$ contains both the GAP relaxation, and the relaxations proposed by Anjos [3] as special cases. Specifically, one obtains the GAP relaxation from $R_{\mathcal{F}}(\phi)$ by taking $\mathcal{F} = \{\alpha \subseteq [n] \,|\, |\alpha| \leq 1\}$. It was proved in [13] that whenever GAP is feasible for a 2-SAT instance, the 2-SAT instance is satisfiable.

The GAP relaxation can be considered as a semidefinite programme in the first level of the well-known Lasserre hierarchy [30]. Anjos [3] proposed semidefinite relaxations of the SAT in approximately levels two and three of the Lasserre hierarchy, by only adding a subset of products of variables to the moment relaxation. For example, in [4], Anjos proposed the R_2 relaxation, which can be obtained from $R_{\mathcal{F}}(\phi)$ by taking

$$\mathcal{F} = \{\alpha \,|\, \alpha \subseteq C_j \text{ for some } j,\, |\alpha| \text{ odd, or } \alpha = \emptyset\}. \tag{13}$$

It was proved in [4] that the R_2 relaxation attains a rank two guarantee on 3-SAT instances: whenever the SDP programme admits a feasible matrix of rank two or lower, the corresponding SAT instance is satisfiable. We will now prove that, for a different $\mathcal{F}$ than (13), the resulting relaxation $R_{\mathcal{F}}(\phi)$ provides the same rank two guarantee.

Theorem 1. *Let ϕ be a 3-SAT instance and*

$$\mathcal{F} = \{\alpha \subseteq [n] \,|\, \alpha \subseteq C_j \text{ for some } j, |\alpha| \neq 1\}. \tag{14}$$

Let Y be the matrix variable of $R_{\mathcal{F}}(\phi)$, indexed by elements of $\mathcal{F}$. If $R_{\mathcal{F}}(\phi)$ admits a feasible rank two matrix, then ϕ is satisfiable.

Proof. The proof is adapted from Theorem 3 in [4], in which the theorem is proven for the case that $\mathcal{F}$ is given as in (13).

Since ϕ is a 3-SAT instance, there exists a clause of length three. Fix a j for which $C_j = \{i_1, i_2, i_3\}$ and set

$$\mathcal{F}_j = \{\alpha \subseteq [n] \,|\, \alpha \subseteq C_j,\, \alpha \in \mathcal{F}\},$$

for $\mathcal{F}$ as in (14). Let Y be feasible solution to $R_{\mathcal{F}}(\phi)$ of rank 2. Consider the submatrix of Y indexed by the elements of $\mathcal{F}_j$,

$$\begin{bmatrix} 1 & Y(i_1i_2) & Y(i_1i_3) & Y(i_2i_3) & Y(i_1i_2i_3) \\ Y(i_1i_2) & 1 & Y(i_2i_3) & Y(i_1i_3) & Y(i_3) \\ Y(i_1i_3) & Y(i_2i_3) & 1 & Y(i_1i_2) & Y(i_2) \\ Y(i_2i_3) & Y(i_1i_3) & Y(i_1i_2) & 1 & Y(i_1) \\ Y(i_1i_2i_3) & Y(i_3) & Y(i_2) & Y(i_1) & 1 \end{bmatrix}. \tag{15}$$

where $Y(\cdot)$ is given as (11). For example, $Y(i_1i_2) = Y_{\emptyset,i_1i_2}$ and $Y(i_1) = Y_{i_2i_3,i_1i_2i_3}$. As the top left 4×4 submatrix of (15) has rank at most 2, it can be proven (Lemma 3.11 from [8]) that at least one of $Y(i_1i_2)$, $Y(i_1i_3)$, $Y(i_2i_3)$ equals $\delta \in \{\pm 1\}$.

Assume without loss of generality that $Y(i_1i_2) = \delta$. Consider now the submatrices of Y indexed by rows and columns $\{\emptyset, i_1i_2, i_1i_3\}$, $\{i_1i_3, i_2i_3, i_1i_2i_3\}$ and $\{\emptyset, i_1i_2, i_1i_2i_3\}$. As the diagonals of these positive semidefinite matrices equal $\mathbf{1}_3$, it can be shown (Lemma 3.9 from [8]) that

$$Y(i_1i_3) = \delta Y(i_2i_3),\ Y(i_2) = \delta Y(i_1),\ Y(i_3) = \delta Y(i_1i_2i_3). \tag{16}$$

This allows us to simplify the satisfiability constraint on C_j, given by

$$1 - a_1Y(i_1) - a_2Y(i_2) - a_3Y(i_3) + a_1a_2Y(i_1i_2) + a_1a_3Y(i_1i_3) + a_2a_3Y(i_2i_3) - a_1a_2a_3Y(i_1i_2i_3) = 0, \tag{17}$$

see (7). Here, we have written a_k for a_{j,i_k}, see (3). We rewrite (17) by substituting $Y(i_1i_2) = \delta$ and (16) to obtain

$$1 - a_1Y(i_1) - a_2\delta Y(i_1) - a_3\delta Y(i_1i_2i_3) + a_1a_2\delta + a_1a_3\delta Y(i_2i_3) + a_2a_3Y(i_2i_3) - a_1a_2a_3Y(i_1i_2i_3) = 0. \tag{18}$$

Using that $a_k^2 = 1$, (18) can be factorized as

$$\Big[1 + a_1a_2\delta\Big]\Big[1 - a_1Y(i_1) + a_2a_3Y(i_2i_3) - a_1a_2a_3Y(i_1i_2i_3)\Big] = 0. \tag{19}$$

In the case $1 + a_1a_2\delta \neq 0$, equation (19) reduces to

$$1 - a_1Y(i_1) + a_2a_3Y(i_2i_3) - a_1a_2a_3Y(i_1i_2i_3) = 0, \tag{20}$$

which is a linear constraint in the entries of matrix Y. In case $1+a_1a_2\delta = 0$, clause C_j holds automatically for $Y(i_1i_2) = \delta$.

Consider the clauses C_j of length three, which can all be factorized as $g_j(\delta)f_j(Y) = 0$, as shown in (19). For all j such that $g_j(\delta) \neq 0$, let B_j be the set of the two subsets appearing in the intersection of $f_j(Y)$ and $\mathcal{F}$. Note that for the particular equation (19), we would have

$$B_j = \{\{i_2, i_3\}, \{i_1, i_2, i_3\}\},$$

in case $1 + a_1a_2\delta \neq 0$. Let B be the union of all B_j. If we define $z_1 = \{i_2, i_3\}$ and $z_2 = \{i_1, i_2, i_3\}$, then (20) is the constraint of the semidefinite relaxation of a clause of length two on the variables z_1 and z_2, as $Y(i_1) = Y_{z_1,z_2}$, see (7).

The submatrix of Y indexed by all sets in B can thus be viewed as a matrix indexed by singletons (by associating a new z variable to each element of B). As Y is feasible for $R_{\mathcal{F}}(\phi)$, it is automatically feasible for the GAP relaxation corresponding to some 2-SAT instance on the z variables, which implies satisfiability of the corresponding 2-SAT instance (Theorem 5.1 [13]). This implies the z variables have a satisfying truth assignment. Given such a truth assignment for the z variables, it is not hard to extend the truth assignment to the original variables of ϕ, by using the appropriate values of δ, see (16). This proves the theorem. □

4 Sum of squares and the MAX-SAT

In Section 4.1 we first provide an overview of the approach of van Maaren et al. [51] for deriving relaxations for the MAX-SAT. Their approach exploits SOS optimization, which has received much attention in the literature, see e.g., [31, 32, 41, 48]. Relaxations depend on a basis that is used to compute them. We introduce a parametric family of monomial bases with increasing complexity. In Section 4.2 we derive several properties of monomial bases that are later used in our computations.

4.1 General overview

For a given logical proposition ϕ, on n variables and m clauses, the value

$$F_\phi(x) := \sum_{j=1}^{m} \frac{1}{2^{\ell_j}} \prod_{i \in C_j} (1 - a_{j,i}x_i), \tag{21}$$

for $a_{j,i}$ as in (3), equals the number of unsatisfied clauses by truth assignment $x \in \{\pm 1\}^n$. Hence, we are interested in minimizing F_ϕ over $\{\pm 1\}^n$, on which F_ϕ is nonnegative. Let $\mathbb{R}[x]$ be the set of real polynomials in x. We define

$$\mathcal{V} := \left\{ f \,\middle|\, f \equiv \sum_{j=1}^{k} f_j^2 \mod I,\ f_j \in \mathbb{R}[x]\ \forall j \in [k],\ k \in \mathbb{N} \right\} \tag{22}$$

as the set of SOS polynomials modulo I, where I is the vanishing ideal of $\{\pm 1\}^n$. That is

$$I = \langle 1 - x_1^2, 1 - x_2^2, \ldots, 1 - x_n^2 \rangle. \tag{23}$$

By *Putinars Positivstellensatz* [45], $\mathcal{V}$ is the set of nonnegative polynomials on $\{\pm 1\}^n$. Generally, optimization over $\mathcal{V}$ is intractable due to its size, which is why we consider

$$\mathcal{V}_{\mathbf{x}} := \{ f \,|\, f \equiv \mathbf{x}^\top M \mathbf{x} \mod I,\ M \succeq 0 \}, \tag{24}$$

where $\mathbf{x}$ is some monomial basis. Since $M \succeq 0$, it follows that all polynomials of $\mathcal{V}_{\mathbf{x}}$ are nonnegative on $\{\pm 1\}^n$. Therefore, $\mathcal{V}_{\mathbf{x}} \subseteq \mathcal{V}$, and we may approximate the minimum of F_ϕ by

$$\min_{x \in \{\pm 1\}^n} F_\phi = \sup\{\lambda \,|\, F_\phi - \lambda \in \mathcal{V}\} \geq \sup\{\lambda \,|\, F_\phi - \lambda \in \mathcal{V}_{\mathbf{x}}\}. \tag{25}$$

The description of $\mathcal{V}_{\mathbf{x}}$ shows that computing this lower bound can be done via SDP.

It is important to note that in the quotient ring of $\mathbb{R}[x]$ modulo I, all terms $x_i^2 \equiv 1$, and thus it suffices to consider only monomials in $\mathbf{x}$ for which the highest power is at most 1. Thus, we can write

$$F_\phi(x) = \sum_{\alpha \subseteq [n]} p_\phi^\alpha x^\alpha, \tag{26}$$

where $p_\phi^\alpha \in \mathbb{R}$ for all $\alpha \subseteq [n]$ and x^α as in (6). For the constant term of $F_\phi(x)$, we have

$$p_\phi^\emptyset = \sum_{j=1}^{m} \frac{1}{2^{\ell_j}}. \tag{27}$$

We say that monomial basis $\mathbf{x}$ *represents* a logical proposition ϕ if matrix $\mathbf{X} \equiv \mathbf{x}\mathbf{x}^\top \mod I$ contains all monomials x^α for which $p_\phi^\alpha \neq 0$. We index this matrix $\mathbf{X}$ and the matrix M from (24) with subsets $\alpha \subseteq [n]$ for which $x^\alpha \in \mathbf{x}$. Note that for such $\alpha, \beta \subseteq [n]$, we have

$$\mathbf{X}_{\alpha,\beta} \equiv x^{\alpha \triangle \beta} \mod I. \tag{28}$$

For $\alpha \subseteq [n]$, we write $x^\alpha \in \mathbf{X}$ if $\mathbf{X}$ has an entry equal to x^α (modulo I). van Maaren et al. [51] propose multiple monomial bases $\mathbf{x}$, among them basis SOS_p, given by

$$\mathbf{x} = 1 \cup \{x_i \,|\, i \in [n]\} \cup \{x_i x_j \,|\, i \text{ and } j \text{ appear together in a clause}\}. \tag{29}$$

It is stated in [51] that SOS_p represents 2-SAT and 3-SAT instances. While this is true, this basis also represents 4-SAT instances (see Lemma 2). We additionally define for $\mathbf{Q} \in \mathbb{N}$, as extension to SOS_p, the basis

$$SOS_p^{\mathbf{Q}} := SOS_p \cup \{x_i x_j \,|\, i \text{ and } j \text{ are both in the top } \mathbf{Q} \text{ appearing variables}\}. \tag{30}$$

Basis $SOS_p^{\mathbf{Q}}$ takes basis SOS_p and adds all the $\binom{\mathbf{Q}}{2}$ quadratic terms of the $\mathbf{Q}$ variables appearing in the highest number of clauses of ϕ. Any basis $\mathbf{x}$ is considered to have duplicate monomials removed, and so, for small values of $\mathbf{Q}$, bases $SOS_p^{\mathbf{Q}}$ and SOS_p might coincide.

We also define the basis SOS_s^θ, for $\theta \in [0, 1]$, which is suited for (MAX-)2-SAT instances. This basis consists of all the monomials of degree one and zero, plus a percentage θ of all quadratic monomials appearing in SOS_p. The included quadratic monomials are those that appear in SOS_p and attain the highest monomial weight w, which is defined as $w(x^\alpha) := \sum_{i \in \alpha} w(i)$, where $w(i) := |\{C \in \phi \,|\, i \in C\}|$, for $i \in [n]$. This results in the following chain of inclusions:

$$\{x^\alpha \,||\alpha| \leq 1\} = SOS_s^0 \subseteq SOS_s^\theta \subseteq SOS_s^1 = SOS_p = SOS_p^0 \subseteq SOS_p^{\mathbf{Q}} \subseteq SOS_p^n = \{x^\alpha \,||\alpha| \leq 2\}. \tag{31}$$

We now define for all $\gamma \subseteq [n]$ such that $x^\gamma \in \mathbf{X}$, a set of ordered pairs as follows

$$\mathbf{x}^\gamma := \{(\alpha, \beta) \subseteq [n]^2 \,|\, \alpha \triangle \beta = \gamma,\, x^\alpha \in \mathbf{x},\, x^\beta \in \mathbf{x}\}. \tag{32}$$

Set $\mathbf{x}^\gamma$ contains the index pairs (α, β) such that $\mathbf{X}_{\alpha,\beta} \equiv x^\gamma$. Therefore, $F_\phi \equiv \mathbf{x}^\top M \mathbf{x}$ if and only if

$$\sum_{(\alpha,\beta)\in\mathbf{x}^\gamma} M_{\alpha,\beta} = p_\phi^\gamma, \quad \forall \gamma \text{ such that } x^\gamma \in \mathbf{X}. \tag{33}$$

Constraints of this form are sometimes referred to as *coefficient matching conditions* in SOS literature [53]. We define

$$\mathcal{M}_\phi := \left\{ M \in \mathcal{S}^{|\mathbf{x}|} \,|\, \sum_{(\alpha,\beta)\in\mathbf{x}^\gamma} M_{\alpha,\beta} = p_\phi^\gamma, \quad \forall \gamma \neq \emptyset \text{ such that } x^\gamma \in \mathbf{X} \right\}, \tag{34}$$

as the set of matrices satisfying the coefficient matching conditions, for all monomials except $x^\emptyset$.

Note that M is constrained to be symmetric which is reflected in the definition of $\mathbf{x}^\gamma$, since $(\alpha, \beta) \in \mathbf{x}^\gamma$ if and only if $(\beta, \alpha) \in \mathbf{x}^\gamma$. Moreover, $\mathbf{x}^\emptyset$ contains the index pairs of the diagonal entries of M, which correspond to zero-degree monomials in $\mathbf{X}$. Hence, if $F_\phi - \lambda \equiv \mathbf{x}^\top M \mathbf{x}$, then $p_\phi^\emptyset - \lambda = \langle I, M \rangle$, see (25) and (27). To maximize the lower bound on F_ϕ, see (25), we maximize λ, which is thus equivalent to minimizing $\langle I, M \rangle$. We can therefore compute this lower bound by solving the following SDP:

$$\begin{aligned} \min \quad & \langle I, M \rangle \\ \text{s.t.} \quad & M \in \mathcal{M}_\phi \cap \mathcal{S}_+. \end{aligned} \tag{$\mathtt{P}_\phi$}$$

We note that, for the purpose of solving $\mathtt{P}_\phi$ through interior point methods, programme $\mathtt{P}_\phi$ is strictly feasible: for any feasible matrix M, matrix $M + I$ is strictly feasible. The existence of any such feasible matrix M follows from the nonnegativity of F_ϕ on $\{\pm 1\}^n$. We postpone the derivation of the dual of $\mathtt{P}_\phi$ to Section 6, where we also show its strict feasibility in Theorem 2.

4.2 Properties of $SOS_p^{\mathbf{Q}}$

We provide provide several properties of monomial bases that are exploited within the PRSM, see Section 7.

Denote by $|\mathbf{x}^\gamma|$ the cardinality of the set $\mathbf{x}^\gamma$, see (32). Due to the symmetry of $\mathbf{X}$, see (28), $|\mathbf{x}^\gamma|$ is an even number and greater than or equal to 2. In particular, when $|\mathbf{x}^\gamma| = 2$, say $\mathbf{x}^\gamma = \{(\alpha, \beta), (\beta, \alpha)\}$, we have

$$M_{\alpha,\beta} + M_{\beta,\alpha} = p_\phi^\gamma \text{ and } M_{\alpha,\beta} = M_{\beta,\alpha} \Longrightarrow M_{\alpha,\beta} = M_{\beta,\alpha} = p_\phi^\gamma/2. \tag{35}$$

Thus, whenever $|\mathbf{x}^\gamma| = 2$, the constraint involving $\mathbf{x}^\gamma$ in $\mathcal{M}_\phi$, see (34), simply fixes two entries of M. van Maaren et al. [51] refer to these constraints arising from $|\mathbf{x}^\gamma| = 2$ as *unit constraints*. In section 7 of [51], the authors empirically show that a high percentage of the constraints of $\mathcal{M}_\phi$ are unit constraints. The authors of [51] propose as future work the development of an SDP solver that is able to exploit the large number of unit constraints. We propose an algorithm for approximately solving $\mathtt{P}_\phi$ in Section 7, which is able to do so.

The following lemma describes the subsets γ that induce unit constraints.

Lemma 1. *Let ϕ be a (MAX-)SAT instance on n variables and m clauses, and $\mathbf{x}$ a monomial basis which contains at least all of the monomials induced by the SOS_p basis, see* (29)*. Then, for all $\gamma \subseteq [n]$, we have*

$$|\mathbf{x}^\gamma| = 2 \quad \Longrightarrow \quad p_\phi^\gamma = 0,$$

where p_ϕ^γ is a coefficient of $F_\phi(x)$, see (26)*.*

Proof. From the definition of $F_\phi(x)$, see (21) and (26), it follows that, for all $\gamma \subseteq [n]$,

$$\gamma \not\subseteq C_j \quad \forall j \in [m] \quad \Longrightarrow \quad p_\phi^\gamma = 0. \tag{36}$$

We proceed by showing that, if $|\mathbf{x}^\gamma| = 2$ for some $\gamma \subseteq [n]$, then γ cannot be contained in a clause. For this purpose, let α be contained in some clause of ϕ, say C_j. We distinguish separate cases for $|\alpha|$. Let

us consider the case $|\alpha| \in \{0,1,2\}$. Then, without loss of generality, we assume $\alpha \subseteq \{i_1, i_2\} \subseteq C_j$. Now (some of) the monomials $x^\alpha \in \mathbf{x}$ induced by C_j are those for which $\alpha \in S := \{\emptyset, i_1, i_2, \{i_1, i_2\}\}$.

We display all the subsets obtained by taking pairwise symmetric differences of elements of S in the symmetric matrix

$$S_\triangle := \begin{bmatrix} \emptyset & i_1 & i_2 & i_1 i_2 \\ & \emptyset & i_1 i_2 & i_2 \\ & & \emptyset & i_1 \\ & & & \emptyset \end{bmatrix}.$$

For clarity, we omitted the lower triangular part of $S_\triangle$ (which is fixed by symmetry). Observe that for any of these α satisfying $|\alpha| \in \{0,1,2\}$, we have $|\mathbf{x}^\alpha| \geq 4$. In the case $|\alpha| = 3$, clause C_j must be of length at least three, and we assume, without loss of generality, that $\alpha = \{i_1, i_2, i_3\} \subseteq C_j$. By again constructing $S_\triangle$ (details omitted), one can show that $|\mathbf{x}^\alpha| \geq 4$. The proof is similar for $|\alpha| = 4$. Lastly, if $|\alpha| = 5$, then by definition of SOS_p, see (29), $|\mathbf{x}^\alpha| = 0$.

Thus, for all $\alpha \subseteq [n]$ contained in some clause of ϕ, $|\mathbf{x}^\alpha| \neq 2$. Therefore,

$$|\mathbf{x}^\gamma| = 2 \Longrightarrow \gamma \not\subseteq C_j \quad \forall j \in [m],$$

which, combined with (36), completes the proof. □

Lemma 1 implies that, in an implementation which uses the SOS_p basis, it is not required to store the coefficients corresponding to unit constraints (since these all equal 0), but only the indices restricted by the unit constraints. The converse of Lemma 1 is generally not true. That is, there can exist many subsets $\gamma \subseteq [n]$ for which $p_\phi^\gamma = 0$, but $|\mathbf{x}^\gamma| > 2$.

Lemma 2. *Let ϕ be a SAT instance on n variables and $\mathbf{x}$ its monomial basis according to $SOS_p^{\mathbf{Q}}$, for some $\mathbf{Q} \in \mathbb{N}$. Let $\gamma \subseteq [n]$. Then*

1. $|\gamma| \in \{1,2\} \Longrightarrow |\mathbf{x}^\gamma| \leq 2n$.
2. $|\gamma| \in \{3,4\} \Longrightarrow |\mathbf{x}^\gamma| \leq 6$.
3. $|\gamma| > 4 \Longrightarrow |\mathbf{x}^\gamma| = 0$.

Proof. The proof follows from enumerating the combination of sets such that their pairwise difference is equal to γ. For part 1 of the lemma, assume first, without loss of generality, that $\gamma = \{1\}$. Then consider the tuples $(\emptyset, \{1\})$ and $(\{1,k\}, \{k\})$, for $k \in [n] \setminus \{1\}$. There are $2n$ of these tuples (counting the symmetry of order twice) and their symmetric differences all equal γ. Next, we assume without loss of generality that $\gamma = \{1,2\}$. Then the tuples $(\{1\}, \{2\})$, $(\{1,2\}, \emptyset)$ and $(\{1,k\}, \{2,k\})$, for $k \in [n] \setminus \{1,2\}$, have their pairwise symmetric difference equal to γ. There are $2n$ of these tuples, which proves part 1 of the lemma.

Assuming that $\gamma = \{1,2,3\}$, we find the 6 tuples as $(\{1\}, \{2,3\})$, $(\{2\}, \{1,3\})$, $(\{3\}, \{1,2\})$. If instead $|\gamma| = 4$, each tuple corresponds to one of $\binom{4}{2} = 6$ partitions which proves part 2.

Lastly, it follows from the definition of $SOS_p^{\mathbf{Q}}$, that any monomial in matrix $\mathbf{x}\mathbf{x}^\top$ is of degree at most four, which proves part 3 of the lemma. □

Part 3 of Lemma 2 shows that the $SOS_p^{\mathbf{Q}}$ bases are only suited for the (MAX-)k-SAT when $k \leq 4$.

5 Resolution and monomial bases

In this section, we consider resolution in combination with the SOS approach to the MAX-SAT. Resolution is a technique from mathematical logic, and widely employed by MAX-SAT solvers [44]. Resolution takes as inputs two clauses of a proposition ϕ, and returns a set of new clauses, named the *resolvent* clauses. The resolvent clauses transform ϕ into ϕ', by either replacing the original clauses, or by adding the resolvent clauses to ϕ (depending on which resolution rule is used). We show in this section that the MAX-SAT resolution rule might not be beneficial for the SOS approach applied to the MAX-SAT, and can even decrease its effectiveness. However, in Section 8.2 we show how to benefit from the SAT resolution rule in the partial MAX-SAT.

We show this at the hand of an example. For $k \geq 3$, we define the following proposition on k variables

$$\phi_k = \begin{cases} \neg x_1 \wedge (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge \neg x_3 & \text{if } k = 3, \\ \neg x_1 \wedge (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge \left[\bigwedge_{j=3}^{k-1} (\neg x_j \vee x_{j+1}) \right] \wedge \neg x_k & \text{else.} \end{cases} \tag{37}$$

It is clear that ϕ_k is unsatisfiable. If one satisfies the initial two unit clauses and performs unit resolution, more unit clauses appear. Repeating this process will lead to an all `false` truth assignment, leaving clause $x_2 \vee x_3$ unsatisfied. Therefore, any truth assignment leaves at least one clause unsatisfied, and hence, $\min_{x \in \{\pm 1\}^k} F_{\phi_k} = F_{\phi_k}(-\mathbf{1}_k) = 1$, for F_{ϕ_k} as in (21).

In the following lemma, we show that the SOS_p basis, see (29), suffices for proving optimality of this assignment.

Lemma 3. *For all $k \geq 3$, we have*

$$\max\{\lambda \,|\, F_{\phi_k} - \lambda \in \mathcal{V}_{\mathbf{x}}\} = 1, \tag{38}$$

where $\mathbf{x} = SOS_p(\phi_k)$, and $\mathcal{V}_{\mathbf{x}}$ as in (24)*.*

Proof. Since $\min_{x \in \{\pm 1\}^k} F_{\phi_k} = 1$, $\nexists \lambda > 1$ such that $F_{\phi_k} - \lambda \in \mathcal{V}$, see (22). As $\mathcal{V}_{\mathbf{x}} \subseteq \mathcal{V}$, this implies that $\nexists \lambda > 1$ such that $F_{\phi_k} - \lambda \in \mathcal{V}_{\mathbf{x}}$. Thus, to prove (38) it suffices to show that $F_{\phi_k} - 1 \in \mathcal{V}_{\mathbf{x}}$, for all $k \geq 3$.

We prove this by induction. For the base case $k = 3$, we have

$$\begin{aligned} F_{\phi_3} &= (6 + x_1 + x_3 - x_1x_2 + x_2x_3)/4 \\ &\equiv 1 + \left[(2 + x_1 + x_3 - x_1x_2 + x_2x_3)^2 + (-x_1 + 2x_2 + x_3 + x_1x_2 + x_2x_3)^2 \right]/32 \mod I. \end{aligned}$$

Clearly, the monomials appearing in the above decomposition are contained in $SOS_p(\phi_k)$. Therefore, $F_{\phi_3} - 1 \in \mathcal{V}_{\mathbf{x}}$. Now for $k + 1$, we have

$$\begin{aligned} F_{\phi_{k+1}} &= F_{\phi_k} + (1 - x_k + x_{k+1} - x_kx_{k+1})/4 \\ &\equiv 1 + (F_{\phi_k} - 1) + (1 - x_k + x_{k+1} - x_kx_{k+1})^2/16 \mod I. \end{aligned}$$

By the induction hypothesis, $F_{\phi_k} - 1 \in \mathcal{V}_{\mathbf{x}}$, and so it follows that $F_{\phi_{k+1}} - 1 \in \mathcal{V}_{\mathbf{x}}$ as well. □

Let us now present the MAX-SAT resolution rule, see e.g., [1]. For clauses C_1 and C_2 of some proposition ϕ, on literals x, z_i, $i \in [s]$ and y_i, $i \in [t]$ construct the clauses below the horizontal line:

$$\begin{gathered} \frac{C_1 = [x \vee z_1 \vee \ldots \vee z_s],\ C_2 = [\neg x \vee y_1 \vee \ldots \vee y_t]}{z_1 \vee \ldots \vee z_s \vee y_1 \vee \ldots \vee y_t,} \\ [C_1 \vee \neg y_1 \vee y_2 \vee \ldots \vee y_t],\ [C_1 \vee \neg y_2 \vee y_3 \vee \ldots \vee y_t], \ldots,\ [C_1 \vee \neg y_t], \\ [C_2 \vee \neg z_1 \vee z_2 \vee \ldots \vee z_s],\ [C_2 \vee \neg z_2 \vee z_3 \vee \ldots \vee z_s], \ldots,\ [C_2 \vee \neg z_s]. \end{gathered} \tag{39}$$

The MAX-SAT resolution rule states that one may replace clauses C_1 and C_2 in ϕ with the $1 + s + t$ resolvent clauses below the horizontal line. We refer to the resulting new proposition, obtained after resolution, as ϕ'. In [10], Theorem 4, it is proven that any truth assignment leaves the same number of clauses unsatisfied for ϕ and ϕ'. This is referred to as *soundness* of the MAX-SAT resolution rule. By soundness and the definition of F_ϕ, see (21), it follows that $F_\phi = F_{\phi'}$.

For standard MAX-SAT solvers, one of the goals of resolution is to create new unit clauses, which are used to compute upper bounds on the MAX-SAT solution [1]. For our SDP approach, if given the same basis, programme $\mathtt{P}_\phi$ equals programme $\mathtt{P}_{\phi'}$, as $F_\phi = F_{\phi'}$. This seems to suggest that MAX-SAT resolution does not change our approach, however, we find that in general $SOS_p(\phi) \neq SOS_p(\phi')$, see (29). We investigate the effect of this difference.

Returning to the example of ϕ_k in (37), let us define $C_q = \neg x_q \vee x_{q+1}$. Observe that for $3 \leq q \leq k-1$, $C_q \in \phi_k$ (assuming $k > 3$). Let us fix some q, $3 \leq q \leq k-3$, and consider the clauses C_q, C_{q+1}, $C_{q+2} \in \phi_k$. We perform resolution as:

$$\frac{C_q = [\neg x_q \vee x_{q+1}],\ C_{q+1} = [\neg x_{q+1} \vee x_{q+2}]}{[\neg x_q \vee x_{q+2}],\ [\neg x_q \vee x_{q+1} \vee \neg x_{q+2}],\ [x_q \vee \neg x_{q+1} \vee x_{q+2}].} \tag{40}$$

We perform resolution again, on the third new clause obtained in (40) and C_{q+2}, to obtain:

$$\begin{gathered} \frac{[x_q \vee \neg x_{q+1} \vee x_{q+2}],\ C_{q+2} = [\neg x_{q+2} \vee x_{q+3}]}{[x_q \vee \neg x_{q+1} \vee x_{q+3}],\ [x_q \vee \neg x_{q+1} \vee x_{q+2} \vee \neg x_{q+3}],} \\ [\neg x_q \vee \neg x_{q+1} \vee \neg x_{q+2} \vee x_{q+3}],\ [x_{q+1} \vee \neg x_{q+2} \vee x_{q+3}], \end{gathered} \tag{41}$$

The resolution rule states that one may replace the original clauses C_1, C_2 and C_3 with the 6 new resolvent clauses obtained from (40) and (41) (the third resolvent from (40) is not counted, since it is replaced in the resolution in (41)).

Observe that the SOS_p basis generates 6 quadratic monomials for the new resolvent clauses, while originally, only 3 quadratic monomials are generated for C_q, C_{q+1} and C_{q+2}. We now define, ϕ'_k for $k \geq 6$, as the logical proposition, obtained by taking ϕ_k, and performing resolution as in (40) and (41), for each triple of clauses $\{C_q, C_{q+1}, C_{q+2}\}$, for each $q \in \{3, 6, 9, \ldots, k-3\}$ (let us assume here that k is a multiple of 3). Note that proposition ϕ'_k constitutes a MAX-4-SAT instance, and therefore basis SOS_p is applicable. Let us compare the sizes of the resulting SOS_p bases, denoted as $|SOS_p|$. We have

$$|SOS_p(\phi_k)| = 2k < 3k - 3 = |SOS_p(\phi'_k)|.$$

Thus, compared to $SOS_p(\phi_k)$, basis $SOS_p(\phi'_k)$ adds approximately k monomials. None of these monomials strengthen the bound, since $SOS_p(\phi_k)$ is already sufficient for proving optimality, by Lemma 3. It is clear that having a larger basis without offering a stronger bound is inefficient, since solving $\mathtt{P}_\phi$ requires more time for larger matrices.

The example of ϕ'_k and ϕ_k shows that not all monomials are (equally) useful in determining bounds. It also shows that resolution can decrease the effectiveness of the SOS approach to the MAX-SAT, by providing 'bad' monomial bases, or it can occur that the SOS_p basis misses 'good' monomials. Our proposed basis $SOS_p^{\mathbf{Q}}$, see (30), attempts to solve this issue.

6 Relating sum of squares and method of moments

In this section, we show how the SOS-SDP relaxation of van Maaren et al. [50, 51] and moment relaxations of Anjos [3]-[6] are related. The relaxations of Anjos, as described in Section 3, were first introduced in 2004 [4] and can be considered as extensions of the GAP relaxation via the well-known Lasserre hierarchy [30]. In 2005, van Maaren et al. [50, 51] proposed the SOS approach to the (MAX-)SAT. Subsequently, van Maaren et al. [51] showed that the SOS relaxation, using monomial basis SOS_{pt} that is larger than SOS_p, see (29), outperforms the R_3 relaxation of Anjos [5], in deciding on the satisfiability of 3-SAT instances. The R_3 relaxation is known to dominate the R_2 relaxation, see (13). In 2007, Anjos [7] strengthened his R_3 relaxation further and left it as future work to determine which SDP relaxation was the strongest.

We complete that work here, by showing a simple relation between the two approaches. In particular, Anjos' relaxations can be considered as method of moments in the Lasserre hierarchy. It is well-known that method of moments is dual to SOS optimization, see [30], and we work out the details here. Let us first derive the dual of the SOS programme $\mathtt{P}_\phi$, see Theorem 2, and then relate it to the here proposed strengthened version of Anjos' relaxations.

To this end, we require the following intermediate result on v_ϕ, see (8).

Lemma 4. *Let $\phi = \bigwedge_{j=1}^m C_j$ be a logical proposition, $v_\phi = \sum_{\alpha \subseteq [n]} v_\phi^\alpha x^\alpha$, see (8), and $F_\phi = \sum_{\alpha \subseteq [n]} p_\phi^\alpha x^\alpha$, see (26). Then,*

$$v_\phi = m - F_\phi.$$

and $v_\phi^\alpha = -p_\phi^\alpha$ for all nonempty $\alpha \subseteq [n]$.

Proof. Let clause C_j have length ℓ_j. We have $v(C_j) = 1 - 2^{-\ell_j} \prod_{i \in C_j}(1 - a_{j,i}x_i)$, see (7). Summing over all clauses yields the desired result. □

Let $\mathbf{x}$ be a given monomial basis, $S \subseteq \mathcal{S}^{|\mathbf{x}|}$. Matrix S is indexed by all $\alpha \subseteq [n]$ for which $x^\alpha \in \mathbf{x}$. To simplify the comparison between the SOS approach and the relaxations of Anjos [3], we define the set

$$\mathcal{X}_\phi := \{S \in \mathcal{S} \,|\, \mathrm{diag}(S) = \mathbf{1},\, S_{\alpha,\beta} = S_{\alpha',\beta'}\ \forall (\alpha,\beta,\alpha',\beta') \subseteq [n] \text{ such that } \alpha \triangle \beta = \alpha' \triangle \beta'\}, \tag{42}$$

for a proposition ϕ on n variables and m clauses. Note that $\mathcal{X}_\phi \subseteq \cap_{j \in [m]} \Delta_j$, see (12), since Δ_j only restricts entries $S_{\alpha,\beta}$ whenever α and β are jointly contained in a single clause. We use $\mathcal{X}_\phi$ in the following theorem.

Theorem 2. *Let ϕ be a logical proposition and $\mathbf{x}$ a monomial basis. The SOS programme $\mathtt{P}_\phi$ defined by ϕ and $\mathbf{x}$, is equivalent to*

$$\begin{aligned} \max \quad & \langle C, S\rangle \\ \text{s.t.} \quad & S \in \mathcal{X}_\phi \cap \mathcal{S}_+, \end{aligned} \tag{43}$$

where $\mathcal{X}_\phi$ is given by (42) and $C \in \mathcal{S}^{|\mathbf{x}|}$, indexed by the subsets $\alpha \subseteq [n]$ for which $x^\alpha \in \mathbf{x}$, is any matrix which satisfies

$$\sum_{(\alpha,\beta)\in\mathbf{x}^\gamma} C_{\alpha,\beta} = v_\phi^\gamma,\ \forall \gamma \neq \emptyset, \mathbf{x}^\gamma \neq \emptyset$$

for v_ϕ^γ as in (8). Moreover, (43) is strictly feasible.

Proof. We rewrite programme $\mathtt{P}_\phi$ by splitting the matrix variable M as follows

$$\begin{aligned} v := \min \quad & \langle I, M\rangle \\ \text{s.t.} \quad & M = Z,\ M \in \mathcal{M}_\phi,\ Z \in \mathcal{S}_+, \end{aligned} \tag{44}$$

where $\mathcal{M}_\phi$ as in (34). We dualize the constraint $M = Z$, and set

$$g(S) := \min_{M\in\mathcal{M}_\phi,\, Z\succeq 0} \langle I, M\rangle + \langle S, M - Z\rangle,$$

for some $S \in \mathcal{S}$. Clearly, $g(S) \leq v$ for all S, and we thus look to maximize $g(S)$, i.e.,

$$\begin{aligned} \max_S g(S) &= \max_S \Big[\min_{M\in\mathcal{M}_\phi} \langle I + S, M\rangle + \min_{Z\succeq 0} \langle S, -Z\rangle \Big] = \max_{S\preceq 0} \ \min_{M\in\mathcal{M}_\phi} \langle I + S, M\rangle \\ &= \max_{S\succeq 0} \ \min_{M\in\mathcal{M}_\phi} \langle I - S, M\rangle. \end{aligned} \tag{45}$$

We now determine the set $\mathcal{X}_\phi$ such that, whenever $S \in \mathcal{X}_\phi$, the minimization over $M \in \mathcal{M}_\phi$ in (45) is bounded. Observe that $\mathcal{M}_\phi$ places no restrictions on the diagonal. To guarantee a bounded minimum, set $\mathcal{X}_\phi$ should restrict $\mathrm{diag}(I - S) = \mathbf{0}$. Each off-diagonal element of a matrix in $\mathcal{M}_\phi$ is restricted by a single constraint of the form (33). Therefore, solving (45) for M can be done by considering separately the elements of M restricted by a single constraint. That is,

$$\begin{aligned} \min_{M\in\mathcal{S}} \quad & \sum_{\gamma\in\mathbf{X}} \sum_{(\alpha,\beta)\in\mathbf{x}^\gamma} -S_{\alpha,\beta} M_{\alpha,\beta} \\ \text{s.t.} \quad & \sum_{(\alpha,\beta)\in\mathbf{x}^\gamma} M_{\alpha,\beta} = p_\phi^\gamma, \end{aligned}$$

where $\mathbf{x}^\gamma$ and $\mathbf{X}$ are defined in (32) and (28), respectively. This minimization problem is bounded if and only if

$$S_{\alpha,\beta} = S_{\alpha',\beta'},\ \forall (\alpha,\beta),\ (\alpha',\beta') \in \mathbf{x}^\gamma\ \forall x^\gamma \in \mathbf{X}, \tag{46}$$

or equivalently, $S_{\alpha,\beta} = S_{\alpha',\beta'}$ for all possible index pairs (α,β) and (α',β') that satisfy $\alpha\triangle\beta = \alpha'\triangle\beta'$. It follows that $\mathcal{X}_\phi$ is given by (42). Now, for fixed $S \in \mathcal{X}_\phi \cap \mathcal{S}_+$, any matrix $M \in \mathcal{M}_\phi$ obtains the same value in (45). Note also that w.l.o.g., we may fix $M = \mathcal{P}_{\mathcal{M}_\phi}(\mathbf{0})$, i.e., the projection of the zero matrix onto $\mathcal{M}_\phi$, (see Lemma 5) which has zero diagonal. This yields the equivalent programme of the form (43), for $C = -M = -\mathcal{P}_{\mathcal{M}_\phi}(\mathbf{0})$. Written explicitly,

$$C_{\alpha,\beta} = -\frac{p_\phi^\gamma}{|\mathbf{x}^\gamma|},\ \forall \alpha,\beta \subseteq [n] \text{ such that } \alpha\triangle\beta = \gamma \text{ (i.e., } (\alpha,\ \beta) \in \mathbf{x}^\gamma).$$

This combined with Lemma 4, proves the claim on matrix C. Lastly, observe that the identity matrix of appropriate size is strictly feasible for (43). □

We define, for $S \in \mathcal{X}_\phi$ and each clause C_j, the function $v^{\mathrm{SDP}}(S, C_j)$, which is obtained by taking (7), and replacing each x^γ by $S_{\alpha,\beta}$, for some $(\alpha,\beta) \in \mathbf{x}^\gamma$. By (46), we are allowed to pick any such (α,β). By Lemma 4, for any nonempty $\gamma \subseteq [n]$, $S \in \mathcal{X}_\phi$ and C as in Theorem 2, we have

$$\sum_{(\alpha,\beta)\in\mathbf{x}^\gamma} C_{\alpha,\beta} S_{\alpha,\beta} = \sum_{(\alpha,\beta)\in\mathbf{x}^\gamma} \frac{-p_\phi^\gamma}{|\mathbf{x}^\gamma|} S_{\alpha,\beta} = -p_\phi^\gamma S_{\alpha,\beta} = v_\phi^\gamma S_{\alpha,\beta}.$$

Hence, maximizing $\langle C, S\rangle$ is equivalent to maximizing the semidefinite relaxation of v_ϕ, see (8), which equals $\sum_{j\in[m]} v^{\text{SDP}}(S, C_j)$.

Moreover, in the relaxations of Anjos [3, 4, 5, 6, 7], outlined in Section 3, the matrix variable is restricted to satisfy $v^{\text{SDP}}(S, C_j) = 1$. Now we can easily observe the difference between the SOS-SDP relaxations and those proposed by Anjos. We present the equivalent dual formulation of the SOS approach below on the left-hand side and the latter (in slightly adapted form) on the right.

$$\begin{aligned} v^* = \max \quad & \sum_{j\in[m]} v^{\text{SDP}}(S, C_j) \\ \text{s.t.} \quad & S \in \mathcal{X}_\phi \cap \mathcal{S}_+. \end{aligned} \tag{47}$$

$$\begin{aligned} \max \quad & 0 \\ \text{s.t.} \quad & S \in \mathcal{X}_\phi \cap \mathcal{S}_+, \\ & v^{\text{SDP}}(S, C_j) = 1, \ \forall C_j. \end{aligned} \tag{48}$$

Note again the difference between (48) and the relaxations described in Section 3, resulting from using set $\mathcal{X}_\phi$ instead of the intersection of the Δ_j, see (12). Thus, we compare the SOS approach with a strengthened variant of the relaxation proposed by Anjos. In Section 8.3, we determine the dual of (48).

Programme (47) proves unsatisfiability of ϕ if $v^* < m$ (with some margin of error, due to numerical precision), while (48) does so whenever the programme is infeasible. The above programmes are not equivalent in this sense: we have empirically found instances ϕ for which $v^* \geq m$, while (48) is infeasible. Neither programme can directly prove satisfiability. However, solutions to both programmes can be used to guide the search towards satisfying assignments (should they exist), see Section 7.3.

If (48) admits a feasible matrix S^*, then matrix S^* is clearly also feasible for (47) and attains an objective value of m. Consequently, in this case, we have $v^* \geq m$. Thus, if (48) does not prove unsatisfiability of ϕ, then neither does (47). In Section 10 we show that (47) can be computed efficiently by applying the PRSM to its dual[1]. It is currently unclear whether a good algorithm for solving (48) exists, and if so, how efficient it would be. Previous numerical experiments on (48) have used general purpose SDP solvers. An immediate improvement might be to use an SDP feasibility problem solver, see [15, 23].

Lastly, the objective value of (47) is more useful for the MAX-SAT: if the underlying instance is infeasible, v^* provides an upper bound to the number of satisfiable clauses, which is useful in a B&B scheme. Programme (48) might also show unsatisfiability of the same instance, but its infeasibility offers no additional value, as to *how* unsatisfiable the instance is.

7 The Peaceman-Rachford splitting method for the MAX-SAT

In this section, we introduce the *Peaceman-Rachford splitting method* [42] for solving SOS-SDP problems and apply it to the MAX-SAT SOS programme $\mathtt{P}_\phi$. Conventionally, interior point methods are used to solve SDP problems. However, for medium and large size instances, interior point methods suffer from a large computation time and memory demand, which has recently motivated researchers to consider first order methods, such as the PRSM. For recent applications of PRSM to SDP, see e.g., [14, 19].

Section 7.2 and Section 7.3 provide details on obtaining valid upper and lower bounds, from the output of the PRSM algorithm.

7.1 The PRSM for SOS relaxations of the MAX-SAT

We start from the reformulation of $\mathtt{P}_\phi$ given in (44). The augmented Lagrangian function of (44) w.r.t. the constraint $M = Z$ for a penalty parameter $\beta > 0$ is:

$$L_\beta(Z, M, S) = \langle I, M\rangle + \langle S, M - Z\rangle + \frac{\beta}{2}\|M - Z\|_F^2.$$

Here, $S \in \mathcal{S}^n$ is the Lagrange multiplier and $\|\cdot\|_F$ denotes the Frobenius matrix norm, see (4).

The PRSM now entails iteratively optimizing over the variables Z and M separately, and updating S twice per cycle. We write superscript k to denote the the value of the variable at iteration k.

$$\begin{cases} Z^{k+1} &= \operatorname{argmin}_{Z\succeq 0} L_\beta(Z, M^k, S^k) &= \mathcal{P}_{\mathcal{S}_+}\left(M^k + \frac{1}{\beta}S^k\right), \\ S^{k+1/2} &= S^k + \gamma_1\beta(M^k - Z^{k+1}), & \\ M^{k+1} &= \operatorname{argmin}_{M\in\mathcal{M}_\phi} L_\beta(Z^{k+1}, M, S^{k+1/2}) &= \mathcal{P}_{\mathcal{M}_\phi}\left(Z^{k+1} - \frac{1}{\beta}\left[I + S^{k+1/2}\right]\right), \\ S^{k+1} &= S^{k+1/2} + \gamma_2\beta(M^{k+1} - Z^{k+1}). & \end{cases} \tag{49}$$

[1] Programme (47) can also be directly solved with the PRSM, as projecting onto $\mathcal{X}_\phi$ is computationally cheap.

Here, $\mathcal{M}_\phi$ is as in (34), and $\mathcal{P}$ is the projection operator as in (5). We have used that

$$\begin{aligned}\operatorname*{argmin}_{Z\succeq 0} L_\beta(Z,M,S) &= \operatorname*{argmin}_{Z\succeq 0} \langle I, M\rangle - \frac{1}{2\beta}\Big\|S\Big\|_F^2 + \frac{\beta}{2}\Big\|Z-(M+\frac{1}{\beta}S)\Big\|_F^2 \\ &= \operatorname*{argmin}_{Z\succeq 0} \frac{\beta}{2}\Big\|Z-(M+\frac{1}{\beta}S)\Big\|_F^2,\end{aligned} \tag{50}$$

see e.g., [39]. In an implementation of (49), one should not store matrix S^k directly, but rather, the matrix $\frac{1}{\beta}S^k$, see Appendix A. When $X \in \mathcal{S}$ has eigenvalues λ_i, and corresponding eigenvectors v_i, it is well known that the projection onto the positive semidefinite cone is given by

$$\mathcal{P}_{\mathcal{S}_+}(X) = \sum_{\{i\,|\,\lambda_i>0\}} \lambda_i v_i v_i^\top = X - \sum_{\{i\,|\,\lambda_i<0\}} \lambda_i v_i v_i^\top. \tag{51}$$

Depending on the number of positive eigenvalues of X, one of the above expressions will be cheaper to compute. The next lemma shows how to compute a projection onto $\mathcal{M}_\phi$.

Lemma 5. *Let matrices M, $\widehat{M} \in \mathcal{S}$, indexed by subsets of $[n]$, such that $\widehat{M} = \mathcal{P}_{\mathcal{M}_\phi}(M)$, where the projection operator $\mathcal{P}_{\mathcal{M}_\phi}(\cdot)$ is given by (5). Then diag$(\widehat{M})$ = diag(M) and*

$$\widehat{M}_{\delta,\mu} = M_{\delta,\mu} - \frac{1}{|\mathbf{x}^\gamma|}\Big(\sum_{(\alpha,\beta)\in\mathbf{x}^\gamma} M_{\alpha,\beta} - p_\phi^\gamma\Big), \tag{52}$$

for $(\delta,\mu) \in \mathbf{x}^\gamma$, $\gamma \neq \emptyset$. In particular, when $|\mathbf{x}^\gamma| = 2$, (52) reduces to

$$\widehat{M}_{\delta,\mu} = \widehat{M}_{\mu,\delta} = p_\phi^\gamma/2. \tag{53}$$

Proof. Let $M \in \mathcal{S}$. To find $\widehat{M} = \mathcal{P}_{\mathcal{M}_\phi}(M)$, note that in $\mathcal{M}_\phi$, each off-diagonal entry is restricted by exactly one constraint. This follows from (34). Since $\mathcal{M}_\phi$ does not restrict the diagonal, it is easily seen that $\operatorname{diag}(\widehat{M}) = \operatorname{diag}(M)$. Now for the off-diagonal entries, we fix a nonempty $\gamma \subseteq [n]$ and define $\mathbf{m}$ as the vector that contains upper triangular entries of M, $M_{\alpha,\beta}$, such that $(\alpha,\beta) \in \mathbf{x}^\gamma$. Similarly, we define $\widehat{\mathbf{m}}$ as the vector containing the same entries of matrix $\widehat{M}$, rather than M. Note that $\mathbf{1}^\top\widehat{\mathbf{m}} = p_\phi^\gamma/2$. Minimizing the Frobenius norm of $\widehat{M} - M$ is now equivalent to minimizing the norm of $\widehat{\mathbf{m}} - \mathbf{m}$. Thus, we solve

$$\widehat{\mathbf{m}} = \operatorname*{argmin}_{\mathbf{1}^\top v = p_\phi^\gamma/2} \|v - \mathbf{m}\|^2,$$

which can be done analytically and leads to (52). The simplification of (52) to (53) follows from the equality $\sum_{(\alpha,\beta)\in\mathbf{x}^\gamma} M_{\alpha,\beta} = 2M_{\delta,\mu}$, whenever $|\mathbf{x}^\gamma| = 2$ and $(\delta,\mu) \in \mathbf{x}^\gamma$. □

Due to the presence of many unit constraints, see (35), these projections are computationally cheap to compute, and hence, the PRSM is well suited to exploit this. Lastly, it is proven [22] that (49) converges for $(\gamma_1, \gamma_2) \in \mathcal{D}$, where

$$\mathcal{D} = \left\{(\gamma_1,\gamma_2)\,\middle|\,\gamma_1+\gamma_2>0,\ |\gamma_1|<1,\ 0<\gamma_2<\frac{1+\sqrt{5}}{2},\ |\gamma_1|<1+\gamma_2-\gamma_2^2\right\}.$$

The values that we choose for (γ_1, γ_2), and other parameters, are given in Section 10.

7.2 Upper bounds and early stopping

After each PRSM iterate k we obtain a triple (Z^k, M^k, S^k) and the resulting $\langle I, M^k\rangle$. Although this value converges to the optimal solution of the SDP, the convergence is (typically) not monotonic and therefore this value does not necessarily provide a valid upper bound for the problem. In this section we describe how to obtain a valid upper bound from the output of the PRSM.

Observe that the feasible set of $\mathtt{P}_\phi$ depends on the chosen monomial $\mathbf{x}$ through $\mathcal{V}_\mathbf{x}$, see (24). Hence, by (25), we have

$$p_\phi^\emptyset - \min_{M\in\mathcal{M}_\phi\cap\mathcal{S}_+} \langle I, M\rangle = \sup_{\lambda\in\mathbb{R}}\{\lambda\,|\,F_\phi - \lambda \in \mathcal{V}_\mathbf{x}\} \leq \min_{x\in\{\pm1\}^n} F_\phi,$$

for $p_\phi^\emptyset$ as in (27). From the above it follows that the maximum number of satisfiable clauses of ϕ is bounded from above by

$$m - p_\phi^\emptyset + \min_{M \in \mathcal{M}_\phi \cap \mathcal{S}_+} \langle I, M \rangle, \tag{54}$$

for m equal to the number of clauses in ϕ. Since the number of satisfied clauses is an integer, the bound (54) can be improved by rounding down the result.

Ideally, the PRSM algorithm (49) computes the upper bound (54) by finding the optimal M in the set $\mathcal{M}_\phi \cap S_+$ (up to some given numerical precision). However, in practice one terminates the PRSM algorithm before this optimal M has been found. Let matrix M^k then be defined as in (49) and let $\lambda_{\min}(M^k)$ be its smallest eigenvalue. Note that

$$\widetilde{M}^k = M^k - \lambda_{\min}(M^k) I \in \mathcal{M}_\phi \cap \mathcal{S}_+, \tag{55}$$

and so, $\widetilde{M}^k$ is feasible for $\mathtt{P}_\phi$. Thus, a valid upper bound at iteration k is obtained as follows:

$$\left\lfloor m - p_\phi^\emptyset + \langle I, \widetilde{M}^k \rangle \right\rfloor. \tag{56}$$

7.3 Lower bounds and rounding

In order to obtain a truth assignment of the variables from the output of the PRSM one needs a rounding procedure. We describe here the rounding procedure proposed by van Maaren et al. [51] and a modification of the procedure that is implemented in our solver.

Let matrix M^* be the optimal solution to the SOS programme $\mathtt{P}_\phi$, induced by a logical proposition ϕ on n variables. Let $\mathbf{x}$ be its monomial basis of size s, and λ^* such that $F_\phi(x) - \lambda^* = \mathbf{x}^\top M^* \mathbf{x}$. It is clear that, by optimality of M^*, $\lambda_{\min}(M^*) = 0$. Let N be the multiplicity of the zero eigenvalue, and $\mathbf{v}_i$, $i \in [N]$ the corresponding eigenvectors. If $y \in \{\pm 1\}^n$ is an optimal MAX-SAT truth assignment of ϕ, then y minimizes F_ϕ. Let y' be the monomial basis vector $\mathbf{x}$, evaluated with the entries of y. Then $y'^\top M^* y' = F_\phi(y) - \lambda^*$. If the SOS relaxation $\mathtt{P}_\phi$ computes the optimal bound, we have $\lambda^* = F_\phi(y)$, which implies that $y' M^* y' = 0$. As the eigenvectors $\mathbf{v}_i$ satisfy the same relation, they can be considered as approximations of maximally satisfying assignments.

Let $V \in \mathbb{R}^{s \times N}$ be the matrix having the vectors $\mathbf{v}_i$, $i \in [N]$ as columns. Each row of V corresponds to a monomial of $\mathbf{x}$. For p the number of monomials in $\mathbf{x}$ of degree two or more, matrix $B \in \mathbb{R}^{p \times N}$ is the submatrix of V obtained by taking the rows of V corresponding to these p monomials. We define $U \in \mathbb{R}^{N \times N}$ as the matrix with columns the eigenvectors of $B^\top B$.

The rounding procedure proposed by van Maaren et al. is to compute $x_\lambda \in \{\pm 1\}^n$ as

$$x_\lambda = \operatorname{sgn}(P_1) \begin{bmatrix} \mathbf{0}_{n \times 1} & I_n & \mathbf{0}_{n \times (s-n-1)} \end{bmatrix} \operatorname{sgn}(P), \text{ for } P = V(U\tilde{\lambda}) \text{ and } \tilde{\lambda}_i = \xi_i \lambda_i \, \forall i \in [N]. \tag{57}$$

Here, $\operatorname{sgn}(\cdot)$ is the sign operator and $\lambda \in \mathbb{R}^N$ a vector generated uniformly at random on the unit sphere (which allows us to perform multiple roundings by generating multiple λ). Observe that P_1, the first entry of vector P, corresponds to the monomial $x^\emptyset$. The vector $\xi \in \mathbb{R}^N$ is a parameter to be chosen. We refer to [51] for the details.

The optimal matrix M^* is a low rank matrix, which ensures that N, the multiplicity of eigenvalue 0, satisfies $N > 1$. In practice however, we do not find M^*, but its approximation $\widetilde{M}^k$ at some iteration, see (55). Due to early stopping, matrix $\widetilde{M}^k$ often has eigenvalue 0 with multiplicity 1. Then λ is a scalar, which does not affect (57). Thus, when $N = 1$, we can only perform one rounding. To solve this issue, we propose constructing V with the columns of q eigenvectors corresponding to the q smallest eigenvalues of $\widetilde{M}^k$ (only in case $N < q$). Thus, whenever $N < q$, we add $q - N$ eigenvectors corresponding to nonzero eigenvalues to the matrix V, in order to perform multiple roundings. In [51], it is observed that the rounding procedure works better when N is small. We use this information by setting $q = 4$, so that we take at least four vectors for the rounding procedure.

8 The weighted partial MAX-SAT

In this section, we extend the SOS approach from the MAX-SAT to the weighted partial MAX-SAT. We also show that the dual formulation of the SOS programme for certain partial MAX-SATs, equals the relaxations by Anjos [5].

In the *weighted* MAX-SAT, each clause is given a weight, and the objective is to maximize the sum of the weights of the satisfied clauses. In the *partial* MAX-SAT, clauses are divided in soft and hard clauses. The aim is to maximize the number of satisfied soft clauses, while satisfying all the hard clauses. The combination of the weighted and partial MAX-SAT is clear, and referred to as the *weighted partial* MAX-SAT [34].

Consider again a logical proposition ϕ. Let $w_j \in \mathbb{R}$ be the weight associated to clause C_j. The generalization of (21) for the (unweighted) MAX-SAT, to the weighted MAX-SAT follows by setting

$$F_\phi^{\mathcal{W}}(x) = \sum_{j=1}^{m} \frac{w_j}{2^{\ell_j}} \prod_{i \in C_j} (1 - a_{j,i} x_i), \tag{58}$$

and then minimizing $F_\phi^{\mathcal{W}}$ for $x \in \{\pm 1\}^n$. This minimization can be approximated by SOS optimization, using directly the semidefinite programme $\mathtt{P}_\phi$.

For the weighted partial MAX-SAT, consider a logical proposition ϕ, on n variables, m soft clauses C_j and q hard clauses $C_p^{\mathbf{H}}$. To each hard clause $C_p^{\mathbf{H}}$, $p \in [q]$, we associate the polynomial $f_p = \prod_{i \in [n]}(1 - a_{p,i} x_i)$, similar to (58). Note that f_p vanishes for all truth assignments that satisfy clause $C_p^{\mathbf{H}}$. Similar to (22) and (24), we define the sets

$$\mathcal{H}_{\mathbf{x}} := \Big\{ \sum_{p \in [q]} c_p f_p \mod I,\, c_p \in \mathbb{R}\ \forall p \in [q] \Big\} \subseteq \mathcal{H} := \Big\{ \sum_{p \in [q]} g_p f_p \mod I,\, g_p \in \mathbb{R}[x]\ \forall p \in [q] \Big\}. \tag{59}$$

Let $\mathbf{SAT} \subseteq \{\pm 1\}^n$ be the set of all truth assignments satisfying the hard clauses (which we assume to be nonempty). From [45] it follows that

$$\min_{x \in \mathbf{SAT}} F_\phi^{\mathcal{W}}(x) = \sup\{\lambda \,|\, F_\phi^{\mathcal{W}} - \lambda \in \mathcal{V} + \mathcal{H}\} \geq \sup\{\lambda \,|\, F_\phi^{\mathcal{W}} - \lambda \in \mathcal{V}_{\mathbf{x}} + \mathcal{H}_{\mathbf{x}}\}, \tag{60}$$

where '+' denotes the Minkowski sum of sets. We proceed by writing the lower bound in (60) as an explicit SDP, for which we introduce the following sets

$$\mathbf{H}^\gamma := \{p \in [q] \,|\, \gamma \in C_p^{\mathbf{H}}\},$$

for $\gamma \subseteq [n]$. Set $\mathbf{H}^\gamma$ contains all p for which f_p, when expanded, contains the term $\pm x^\gamma$. The sign here is determined by the parity of $|\gamma \cap I_p^+|$, see (2). Additionally, we define as analogue to $\mathcal{M}_\phi$, see (34), the set $\mathcal{M}_\phi^{\mathbf{H}}$. This set contains all matrices M and vectors c such that $F_\phi^{\mathcal{W}} - \lambda \equiv \mathbf{x}^\top M \mathbf{x} + \sum_{p \in [q]} c_p f_p \mod I$. It is therefore defined as

$$\mathcal{M}_\phi^{\mathbf{H}} := \Big\{ (M, c) \in \mathcal{S} \times \mathbb{R}^q \,|\, \sum_{(\alpha,\beta) \in \mathbf{x}^\gamma} M_{\alpha,\beta} + \sum_{p \in \mathbf{H}^\gamma} (-1)^{|\gamma \cap I_p^+|} c_p = p_\phi^\gamma,\ \forall \gamma \neq \emptyset \text{ such that } x^\gamma \in \mathbf{X} \Big\}. \tag{61}$$

This allows us to adapt $\mathtt{P}_\phi$ to the weighted partial MAX-SAT as follows:

$$\begin{aligned} \min \quad & \langle I, M \rangle + \sum_{p \in [q]} c_p \\ \text{s.t.} \quad & (M, c) \in \mathcal{M}_\phi^{\mathbf{H}},\ M \in \mathcal{S}_+. \end{aligned} \tag{62}$$

We approximately solve (62) by the PRSM, see Section 8.1. Let us elaborate on how to adapt the monomial bases to the (weighted) partial MAX-SAT. We make no distinction between soft and hard clauses for the SOS_p basis, see (29). For basis SOS_s^θ, see (31), we determine the variable weights as $w(i) := \sum_{\{j \,|\, i \in C_j\}} w_j + \sum_{\{p \,|\, i \in C_p^{\mathbf{H}}\}} \overline{w}$, for $\overline{w}$ the mean of all soft clause weights w_j. For basis $SOS_p^{\mathbf{Q}}$, we add all $\binom{\mathbf{Q}}{2}$ quadratic terms of the $\mathbf{Q}$ variables that attain the highest value of $w(i)$. For unweighted partial MAX-SAT instances, we consider all w_j to equal 1.

8.1 The PRSM for SOS of the weighted partial MAX-SAT

We show here how to solve (62) by the PRSM. We first rewrite (62) by introducing the matrix variable Z, see also (44),

$$\begin{aligned} \min \quad & \langle I, M \rangle + \sum_{p \in [q]} c_p \\ \text{s.t.} \quad & M = Z,\ (M, c) \in \mathcal{M}_\phi^{\mathbf{H}},\ Z \in \mathcal{S}_+. \end{aligned} \tag{63}$$

Then, the augmented Lagrangian function of (63) w.r.t. $Z = M$ and for a penalty parameter $\beta > 0$ is:

$$L_\beta(Z, M, S, c) = \langle I, M\rangle + \langle S, M - Z\rangle + \frac{\beta}{2}\|M - Z\|_F^2 + \mathbf{1}^\top c.$$

The PRSM is iteratively and separately optimizing over $(M, c) \in \mathcal{M}_\phi^{\mathbf{H}}$ and $Z \in \mathcal{S}_+$, and updating S twice per cycle, similarly to (49). However, in this case, the M-subproblem from (49) is replaced by the (M, c)-subproblem.

We now show that minimization over $(M, c) \in \mathcal{M}_\phi^{\mathbf{H}}$ can be performed efficiently. By derivations similar to (50), we have:

$$\operatorname*{argmin}_{(M,c)\in\mathcal{M}_\phi^{\mathbf{H}}} L_\beta(Z, M, S, c) = \operatorname*{argmin}_{(M,c)\in\mathcal{M}_\phi^{\mathbf{H}}} \|M - X\|_F^2 + \frac{2}{\beta}\mathbf{1}^\top c, \tag{64}$$

where $X := Z - (S + I)/\beta$. This is a convex quadratic programme (QP) that we solve in two steps. Firstly, consider the matrix-entries $M_{\alpha,\beta}$, with $(\alpha, \beta) \in \mathbf{x}^\gamma$, see (32), and $\mathbf{H}^\gamma = \emptyset$. Since $\mathbf{H}^\gamma = \emptyset$, these entries are unaffected by the c_p variables. This implies that these $M_{\alpha,\beta}$ variables are not coupled with the other entries of M, and one can minimize separately over such $M_{\alpha,\beta}$. This separate minimization problem can be solved by applying Lemma 5.

Secondly, the remaining QP

$$\min \sum_{\{\gamma \,|\, \mathbf{H}^\gamma \neq \emptyset\}} \sum_{(\alpha,\beta)\in\mathbf{x}^\gamma} (M_{\alpha,\beta} - X_{\alpha,\beta})^2 + \frac{2}{\beta}\mathbf{1}^\top c, \tag{65}$$

can be simplified by the following observation. If M^* is an optimal solution to (64), then

$$(\alpha, \beta), (\alpha', \beta') \in \mathbf{x}^\gamma \implies M^*_{\alpha,\beta} - X_{\alpha,\beta} = M^*_{\alpha',\beta'} - X_{\alpha',\beta'}.$$

Hence, (65) can be simplified by substituting each term $\sum_{(\alpha,\beta)\in\mathbf{x}^\gamma}(M_{\alpha,\beta} - X_{\alpha,\beta})^2$ with a single squared variable. We solve the resulting QP either by solving the KKT conditions using the LU decomposition, or via MOSEK [38]. The solving method depends on the underlying QP.

8.2 Strengthening the bounds

We demonstrate a simple technique for improving the upper bounds given by programme (62). This technique is based on the SAT resolution rule, which is given as follows. For two hard clauses of some proposition ϕ, on literals x, z_i, $i \in [s]$ and y_i, $i \in [t]$, construct the clause below the horizontal line:

$$\frac{[x \vee z_1 \vee \ldots \vee z_s],\ [\neg x \vee y_1 \vee \ldots \vee y_t]}{z_1 \vee \ldots \vee z_s \vee y_1 \vee \ldots \vee y_t.} \tag{66}$$

In contrast to the MAX-SAT resolution rule (39), the SAT resolution rule states that one may add the clause below the horizontal line to ϕ, without changing its (un)satisfiability (we say that the new clause is *implied* by the original two clauses). We may apply this SAT resolution rule to the hard clauses of a partial MAX-SAT instance to generate more hard clauses. As each new clause induces a new variable c_p, the bound of programme (62) can only improve. One may also regard SAT resolution as extending the set $\mathcal{H}_{\mathbf{x}}$, see (59), by including terms of the form $c_p x^\alpha f_p$, for some $\alpha \subseteq [n]$ where $c_p \in \mathbb{R}$.

Additionally, SAT resolution can generate hard unit clauses. This is advantageous, since hard unit clauses reduce the number of variables in the MAX-SAT, see Section 1.2.

8.3 Duality in the partial MAX-SAT

Now we consider a partial MAX-SAT with only hard clauses. Solving such instances is thus equivalent to determining the satisfiability of the given hard clauses. We show that by taking the dual of the resulting SOS programme, one obtains (a stronger version of) the relaxations of Anjos [4], given by (48).

We define, for $A \in \mathcal{S}^n$, $\operatorname{vec}(A) \in \mathbb{R}^{n^2}$ the vector whose entries are the columns of A stacked together. We start from programme (62) and perform variable splitting on M, similar to (44). We take the dual $g(S)$ of this formulation, similar to (45), and consider the problem

$$\max_S g(S) = \max_{S\in\mathcal{X}_\phi\cap\mathcal{S}_+} \min_{(M,c)\in\mathcal{M}_\phi^{\mathbf{H}}} \langle S, -M\rangle + \mathbf{1}^\top c, \tag{67}$$

for $\mathcal{X}_\phi$ as in (42). The steps that show that $S \in \mathcal{X}_\phi \cap \mathcal{S}_+$ is necessary for the above expression to be finite are provided in the proof of Theorem 2. We rewrite the inner minimization problem in (67) as

$$\min_{(M,c)\in\mathcal{M}_\phi^{\mathbf{H}}} \begin{bmatrix} \mathrm{vec}(S) \\ \mathbf{1} \end{bmatrix}^\top \begin{bmatrix} \mathrm{vec}(-M) \\ c \end{bmatrix}, \tag{68}$$

and proceed to show under which conditions this value is bounded. Observe that the coefficients $p_\phi^\gamma = 0$, see (61), since there are no soft clauses. Moreover, the set $\mathcal{M}_\phi^{\mathbf{H}}$ places only linear constraints on the entries of M and c. Therefore, there exists a matrix D that satisfies

$$(M,c) \in \mathcal{M}_\phi^{\mathbf{H}} \iff D \begin{bmatrix} \mathrm{vec}(-M) \\ c \end{bmatrix} = \mathbf{0}.$$

Hence, (68) is bounded if and only if $\begin{bmatrix}\mathrm{vec}(S)^\top & \mathbf{1}^\top\end{bmatrix}$ is contained in the row space of D. This is precisely the requirement that $v^{\mathrm{SDP}}(S, C_p^{\mathbf{H}}) = 1$, $\forall p \in [q]$, as in (48). We provide one example of this claim.

Example 1. *Consider the monomial basis* $\mathbf{x} = (x^\emptyset, x_1, x_2)$. *Let* $C_1^{\mathbf{H}} = x_1 \vee x_2$, *so that* $f_1 = 1 - x_1 - x_2 + x_1x_2$. *Now*

$$(M,c) \in \mathcal{M}_\phi^{\mathbf{H}} \Longrightarrow Du = \mathbf{0}, \text{ for } D = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 1 \end{bmatrix},$$
$$\text{and } u = \begin{bmatrix} -M_{1,\emptyset} & -M_{\emptyset,1} & -M_{2,\emptyset} & -M_{\emptyset,2} & -M_{1,2} & -M_{2,1} & c_1 \end{bmatrix}^\top.$$

For $S \in \mathcal{X}_\phi$, *and by definition of* $\mathcal{X}_\phi$ (42), *we have* $S_{1,\emptyset} = S_{\emptyset,1}$, *and similar equalities hold for all other related entries of matrix* S. *Thus, we may remove duplicate columns in* D. *We have*

$$\begin{bmatrix} S_{1,\emptyset} & S_{2,\emptyset} & S_{1,2} & 1 \end{bmatrix} \in \mathrm{row} \begin{bmatrix} -1 & 0 & 0 & -1 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \Longrightarrow S_{1,\emptyset} + S_{2,\emptyset} - S_{1,2} = 1 \Longrightarrow v^{\mathrm{SDP}}(S, C_1^{\mathbf{H}}) = 1.$$

Thus, (68) is bounded if $\begin{bmatrix}\mathrm{vec}(S)^\top & \mathbf{1}^\top\end{bmatrix} \in \mathrm{row}(D)$, in which case, the value equals zero. Hence, programme (67) is equivalent to (48).

9 SOS-MS: Algorithm description

In this section, we elaborate on the algorithm behind our complete SOS-SDP based MAX-SAT solver, named SOS-MS. In particular, we outline the main parts of SOS-MS and provide a pseudocode, see Algorithm 1.

Consider a given MAX-k-SAT instance, $k \leq 3$, and corresponding logical proposition ϕ. SOS-MS uses the PRSM, see (49), to obtain an approximate solution $\widetilde{M}$, see (55), to $\mathtt{P}_\phi$. Then, using (56), SOS-MS determines an upper bound `UB` and a lower bound on the optimal value, by applying the rounding procedure from Section 7.3. The solver calls once, at the beginning, the CCLS[2] algorithm [35] for the MAX-SAT, to compute a lower bound. CCLS is a local search algorithm whose performance was one of the best among tested heuristic algorithms in the MSE-2016. We set `LB` as the maximum value attained by these two methods.

In case `LB` = `UB`, we have proven optimality and the algorithm terminates. In case `LB` < `UB`, we branch on some variable x_i, $i \in [n]$, by assigning it either `true` or `false`. This resembles to performing unit resolution, see Section 1.2. We write $\phi' = \mathtt{unitRes}(\phi, i)$ to indicate that ϕ' is the logical proposition obtained from ϕ by setting $x_i = 1$ (equivalently, $x_i = \mathtt{true}$). We use the same notation to indicate the logical proposition obtained from ϕ by setting $x_i = -1$ (equivalently, $x_i = \mathtt{false}$), i.e., $\phi' = \mathtt{unitRes}(\phi, -i)$. To emphasize the difference between ϕ and ϕ', in this section, we write n_ϕ and m_ϕ for the number of variables and clauses of ϕ.

If we branch on x_i, we remove from the monomial basis $\mathbf{x}$ all monomials x^α that satisfy $i \in \alpha$. We remove from matrices Z^k, M^k and S^k, that were obtained in the last relevant call of the PRSM, all rows and columns corresponding to such subsets α. The resulting matrices are then used as the new Z^0, M^0 and S^0 in the next PRSM call, i.e., those are used as a warm start.

[2]The CCLS algorithm is publicly available at `http://lcs.ios.ac.cn/~caisw/MaxSAT.html`.

We determine the order of variables for branching as follows. First, we consider for $b \in \{-n, \dots, n\} \setminus \{0\}$, the values

$$u_b = m_{\phi'} - p^{\emptyset}_{\phi'} + \langle I, \widetilde{M} \rangle - \sum_{|b| \in \alpha} \widetilde{M}_{\alpha,\alpha}, \text{ for } \phi' = \texttt{unitRes}(\phi, b), \tag{69}$$

similar to (54). Here, $\widetilde{M}$ is the approximate solution to $\texttt{P}_\phi$. We remark that (69) can be quickly computed without explicitly performing the unit resolution. Observe that $\langle I, \widetilde{M} \rangle - \sum_{|b| \in \alpha} \widetilde{M}_{\alpha,\alpha}$ equals the trace of the new matrix M^0, which is used as warm start for $\texttt{P}_{\phi'}$.

Second, we perform the rounding procedure to $\widetilde{M}$ as described in Section 7.3. That is, we randomly generate a set Λ of vectors drawn uniformly at random on the unit sphere, and compute the corresponding rounded truth assignments $x_\lambda \in \{\pm 1\}^{n_\phi}$, $\lambda \in \Lambda$, by (57). We update LB if a better truth assignment is found. Let $\Lambda^* \subseteq \Lambda$ contain all the vectors λ that satisfy $F_\phi(x_\lambda) = m_\phi - \texttt{LB}$, and, assuming $\Lambda^* \neq \emptyset$, set $\mathbf{v} := \frac{1}{|\Lambda^*|} \sum_{\lambda \in \Lambda^*} x_\lambda$. Note that $-1 \leq \mathbf{v}_i \leq 1 \ \forall i \in [n]$ with equality if and only if x_i is assigned the same truth value for all x_λ, $\lambda \in \Lambda^*$. We define

$$B := \left\{ b \mid 0 < |b| \leq n_\phi, \ b \in \mathbb{Z}, \ \mathbf{v}_{|b|} = -\text{sgn}(b) \right\},$$

and explain its purpose by an example. If $-3 \in B$, then x_3 is assigned `true` by all x_λ, $\lambda \in \Lambda^*$. Heuristically, branching by setting $x_3 = -1$ would then hopefully lead to low upper bounds in the resulting search tree. This is advantageous because low upper bounds lead to faster pruning. In case $\Lambda^* = \emptyset$, we set $B = \{-n, \dots, n\} \setminus \{0\}$.

Lastly, for all $b \in B$, we sort them in increasing order of u_b, see (69), and store this order in vector σ. Thus, the entries of σ satisfy

$$u_{\sigma_j} \leq u_{\sigma_{j+1}} \text{ and } \sigma_j \in B. \tag{70}$$

Vector σ determines the variable selection in the branching process of SOS-MS, which we describe in more detail in the sequel.

Consider a node in the SOS-MS search tree, in which we consider the proposition ϕ. We initialize $b_* := 0$. For increasing values of $j \in [b_{\max}]$, where $b_{\max} > 1$ is some integer (see (71)), we compute the SDP upper bound corresponding to $\texttt{unitRes}(\phi, \sigma_j)$, denoted UB. In case $\lfloor \texttt{UB} \rfloor \leq \texttt{LB}$, we repeat this process with the next value of σ_j, and update $b_* := b_* + 1$. In case $\lfloor \texttt{UB} \rfloor > \texttt{LB}$, we terminate the process.

In case $b_* > 0$, we find that for all $j \leq b_*$, the propositions $\texttt{unitRes}(\phi, \sigma_j)$ cannot improve on LB. Thus, we may limit the search for better truth assignments to $\texttt{unitRes}(\phi, -\sigma_1, \dots, -\sigma_{b_*})$. In case $b_* = 0$, we add both $\texttt{unitRes}(\phi, \sigma_1)$ and $\texttt{unitRes}(\phi, -\sigma_1)$ to the search tree, as we cannot exclude either one from attaining a value strictly greater than LB.

We take the previously mentioned $b_{\max}$ as

$$b_{\max} = \min\left\{ \max\left\{ 3; \lfloor 6\,\texttt{GAP} + 1/2 \rfloor \right\}; 6 \right\}, \text{ for } \texttt{GAP} = m_{\phi'} - p^{\emptyset}_{\phi'} + \langle I, \widetilde{M} \rangle - \texttt{LB} - 1, \tag{71}$$

where $\widetilde{M}$ is an approximate solution to $\texttt{P}_\phi$.

A pseudocode of SOS-MS is given by Algorithm 1. In particular, the branching process is described in Lines 15 to 20. Note the two PRSM calls in Lines 8 and 16. In Line 8, the main purpose of the PRSM is to find an upper bound which equals the best known lower bound. When this does not occur, we use the approximate solutions as warm start for the PRSM call in Line 16. In Line 16, the purpose of the PRSM is to prune the node corresponding to ϕ'. We use the LOBPCG algorithm [28] to efficiently approximate $\lambda_{\min}(M^k)$, which allows us to compute approximate upper bounds during the PRSM iterates, see (54) and (55). In case the approximate upper bound indicates that the node can be pruned, we recompute $\lambda_{\min}(M^k)$ with the more accurate MATLAB `eig` function.

The algorithm can then stop iterating as soon as the condition in Line 17 is satisfied, or when it is clear that this condition cannot be satisfied in reasonable time.

Remark 1. *Most of the running time of SOS-MS is spent on computing projections of matrices onto $\mathcal{S}_+$, see (51). We found that computing the full spectra of the matrices to be projected (in single-precision, rather than standard double-precision) trough MATLAB's `eig` command was the fastest method of computing $\mathcal{P}_{\mathcal{S}_+}(\cdot)$, even though only the positive eigenpairs, or negative eigenpairs are required. For a variant of the PRSM, the authors of [47] propose using the LOBPCG algorithm [28] to compute only positive/negative eigenpairs (when this number is deemed small enough) of matrices to be projected. We were unable to obtain a speedup over `eig` through this method in the PRSM framework.*

Algorithm 1: SOS-SDP MAX-SAT Solver

Input: A MAX-k-SAT instance ϕ ($k \leq 3$), on n_ϕ variables and m_ϕ clauses.
Output: Optimal truth assignment $x \in \{\pm 1\}^n$.
Set $\texttt{UB} := m_\phi$.
Use the CCLS algorithm on ϕ to obtain a value for current best lower bound LB and corresponding truth assignment $x \in \{\pm 1\}^n$.
Initialize the stack $Q := (\phi, \texttt{UB})$.
while $Q \neq \emptyset$ **do**
    Take $(\phi, \texttt{UB})$ as the first element of Q.
    Use the PRSM, see (49), to obtain an approximate solution $\widetilde{M}$, see (55), to $\texttt{P}_\phi$.
    Apply the rounding procedure from Section 7.3 to $\widetilde{M}$ and obtain a lower bound LB. Update LB and x if a better truth assignment has been found.
    Set $\texttt{UB} := \min\{\texttt{UB}, \lfloor m_\phi - p_\phi^\emptyset + \langle I, \widetilde{M} \rangle \rfloor\}$, see (54).
    Remove $(\phi, \texttt{UB})$ from the stack Q.
    if $\texttt{LB} \geq \texttt{UB}$ **then**
        /* The current node $(\phi, \texttt{UB})$ can be pruned, so return to Line 6, and check if Q is empty. */
        continue
    Determine σ, see (70), and $b_{\max}$, see (71). Set $b_* := 0$.
    for $j = 1$ `to` $b_{\max}$ **do**
        Set $\phi' := \texttt{unitRes}(\phi, \sigma_j)$, use the PRSM to obtain an approximate solution $\widetilde{M}$ to $\texttt{P}_{\phi'}$.
        /* Use as warm start the matrices obtained from the programme $\texttt{P}_\phi$ in Line 8. */
        if $\texttt{LB} \geq \lfloor m_{\phi'} - p_{\phi'}^\emptyset + \langle I, \widetilde{M} \rangle \rfloor$ **then**
            $b_* := b_* + 1$.
        else
            break
    if $b_* > 0$ **then**
        Set $\phi := \texttt{unitRes}(\phi, -\sigma_1, -\sigma_2, \ldots, -\sigma_{b_*})$, and $Q := Q \cup (\phi, \texttt{UB})$.
        /* Q is nonempty, so return to Line 6. */
    else
        Set $\phi_1 := \texttt{unitRes}(\phi, \sigma_1)$, $\phi_2 := \texttt{unitRes}(\phi, -\sigma_1)$ and $Q := Q \cup \{(\phi_1, \texttt{UB}), (\phi_2, \texttt{UB})\}$.
        /* Q is nonempty, so return to Line 6. */
return Optimal truth assignment x.

9.1 Parsing sum of squares programmes

We cover here the problem of initializing an SOS semidefinite programme. Methods for achieving this are built in most SOS packages, such as SOSTOOLS [40] and GlobtiPoly [24]. In our application, we are interested in SOS modulo a vanishing ideal, which is not natively implemented in most SOS software, but rather, by restriction of the support set of the variables to a semialgebraic set (such as $x^2 = 1$), which incurs additional variables in the semidefinite programme.

For most SDP applications, it is implicitly assumed that the programme parameters are either already given, or require a negligible time to compute, in comparison to the time required for solving the resulting semidefinite programme. For most SOS programmes however, this is decidedly not the case. The authors of SOSTOOLS [40], a third-party MATLAB package for formulating and solving SOS programmes, confirm this observation. In chapter one of the user's manual to SOSTOOLS [40], it is stated that defining the semidefinite programme, rather than solving it, is often the limiting factor for tractable problem size. In accordance with [40], we thus consider the problem of *parsing* an SOS programme as defining it by its programme parameters. This definition of parsing includes the problem of choosing the monomial basis $\mathbf{x}$ such that the given polynomial can be accurately described. Many theoretical results can guide the choice of $\mathbf{x}$, such as the Newton polytope [49], see also [46], or facial reduction [43]. For our purposes however, choosing $\mathbf{x}$ can be done with a simple fixed procedure, see e.g., (29). We therefore consider parsing as only the purely numerical problem of finding, rather than choosing, $\mathbf{x}$.

In this section, we provide a short overview of our parsing method. We exploit the fact that our variables are $\{\pm 1\}^n$, which allows us to achieve fast parsing times, compared to general purpose SOS software. For example, due to the properties of computation modulo I, see (23), monomials can be stored in two ways: either we store some $\alpha \subseteq [n]$, corresponding to x^α as in (6), (*subset format*) or we store monomials as a (possibly sparse) vector $\mathbf{v} \in \{0,1\}^n$, corresponding to $x^{\mathbf{v}} = x_1^{v_1} \dots x_n^{v_n}$ (*vector format*). Such vectors can be saved as sparse boolean vectors. It is trivial to switch between these two formats, which is what we use in our parsing algorithm: we implement each step using the best suited format. Also note that for monomial basis $\mathbf{x}$ given by (29), monomials in $\mathbf{X} \equiv \mathbf{x}\mathbf{x}^\top$, see (28), will have degree at most 4, which ensures that both formats require little storage.

Let ϕ be the considered proposition, on n variables and m clauses. Initially, to compute $\mathbf{x}$ according to (29), we consider the unique clauses C_j of ϕ, $j \in [m]$. Recall that we define a clause C_j as a subset of $[n]$, see Section 1.1. To compute the monomials in $\mathbf{X}$, we need to compute the cross products of all monomials in $\mathbf{x}$. This is best done in vector format, by using the entrywise *exclusive or* operation on boolean vectors, denoted $\oplus$. That is, $x^{\mathbf{v}} x^{\mathbf{u}} \equiv x^{\mathbf{v} \oplus \mathbf{u}} \mod I$, which is computationally cheaper than the symmetric difference operator as in (28).

Next, to determine the sets $\mathbf{x}^\gamma$, as in (32), we need to find the sets of equal monomials in $\mathbf{X}$. We first split up the monomials in groups based on their degree, which is trivially computed in vector format as $\mathbf{v}^\top \mathbf{1}$. Then for the monomials of degree one, we switch to subset format and find the groups of equal monomials based on these one element subsets. For degrees two and three, this procedure is similar, except we further divide these groups in n smaller subgroups, based on what the first nonzero entry in their vector $\mathbf{v}$ is. If we consider the SOS_p basis, see (29), then for non-trivially sized instances, most of the monomials in $\mathbf{x}\mathbf{x}^\top$ will be of degree 4. Therefore, we divide the monomials of degree 4 in n^2 subgroups, based on the first and second nonzero entry in their vector $\mathbf{v}$. Then, in each of these subgroups, we switch to subset format, which yields a matrix of four columns, and each row associated to a single monomial. The first two columns of this matrix are fixed by which subgroup we consider; hence we only evaluate the third and fourth columns of this matrix. In these columns, we then search for unique rows, which represent unique monomials.

We compute the coefficients p_ϕ^γ, see (26), iteratively per clause. If k is the size of the largest clause in ϕ, we create $k+1$ matrices A_s, for $0 \leq s \leq k$, where matrix A_s is an s-dimensional matrix. Then, for $\gamma = \{\gamma_1, \dots, \gamma_s\}$, we store p_ϕ^γ at position $(A_s)_{\gamma_1,\dots,\gamma_s}$. Note that matrix A_0 is a number storing $p_\phi^\emptyset$, see (27). For (MAX-)3-SAT instances, matrices A_1 and A_2 will be full, matrix A_3 will (generally) be sparse, and matrix A_4, although we do not create it, would be empty. We have also tested the recently developed `dpvar` structure of SOSTOOLS [25] to compute the p_ϕ^γ symbolically, but found that it compared unfavourably to our procedure, in terms of computation time.

The last step is to match the coefficients p_ϕ^γ to the monomial x^γ of $\mathbf{X}$. Clearly, we need only to consider those coefficients p_ϕ^γ for which $p_\phi^\gamma \neq 0$. Thus, for some nonzero p_ϕ^γ stored in A_s, we know that $|\gamma| = s$. As the monomials in $\mathbf{X}$ have already been divided into groups based on their degree, we search only the monomials of degree s, to find x^γ. In case we use the SOS_p basis, then by Lemma 1, we need only check those monomials x^α for which $|\mathbf{x}^\alpha| > 2$, see (32). Now to find x^γ in this subgroup of monomials (of which one equals x^γ), we use the vector format. By exploiting properties of the ideal I, see (23), and SOS_p, we are able to obtain high parsing speeds. For example, instance `s3v70c1500-1.cnf` (70 variables and 1500 clauses of length 3) from the MSE-2016[3], induces an SOS_p basis of size 2107. Matrix $\mathbf{X}$, see (28), then contains 4,439,449 monomials. Our algorithm parses this basis in (approximately) 1 second.

This completes the summary of our parsing algorithm. For more details, interested readers are referred to the code available on `GitHub`[4].

10 Numerical results

In this section, we test SOS-MS, described in Section 9, on MAX-3-SAT, weighted partial MAX-2-SAT, and weighted MAX-3-SAT instances from the MSE-2016. We use instances from the same source to compute bounds for the partial MAX-3-SAT. We choose the year 2016, because later years of the MSE offer no MAX-2-SAT or MAX-3-SAT instances. The instances in this section are taken from the MSE-2016[5]random track.

[3] All instances are available at `http://www.maxsat.udl.cat/16/benchmarks/index.html`.
[4] Code available at `https://github.com/LMSinjorgo/SOS-SDP_MAXSAT`.
[5] For more information on the MSE-2016, see `http://maxsat.ia.udl.cat/introduction/`.

Experiments are carried out on a 16 GB RAM laptop, with an Intel Core i7-1165G7 (2.8 GHz) and four cores, running Windows 10 Enterprise. We set the PRSM parameters, see (49), as

$$(\gamma_1, \gamma_2, \beta) = \Big(\frac{1}{2}, \frac{9}{10}\frac{1+\sqrt{5}}{2}, \frac{2s}{5}\Big),$$

where s is the order of the matrix variables. Our MATLAB implementation of the PRSM is available at `https://github.com/LMSinjorgo/SOS-SDP_MAXSAT`.

10.1 MAX-3-SAT instances

In this section we first show a relation between upper bounds obtained by solving $\mathtt{P}_\phi$ and $SOS_p^{\mathbf{Q}}$ bases (30). Then, we demonstrate the performance of SOS-MS.

Table 1 presents a comparison of the bounds obtained by solving the SOS programme $\mathtt{P}_\phi$ using the SOS_p basis (29) and the $SOS_p^{\mathbf{Q}}$ bases (30) where $\mathbf{Q} \in \{40, 50, 60, 70, 110\}$, for several instances in category MAX-3-SAT on 70, 90 and 110 variables. The first column reports the name of the corresponding instance, on `v` variables and `c` clauses. Column `LB` provides the best found lower bound, obtained by running the CCLS algorithm [35] for five seconds. Column `UB` reports the computed upper bounds. Column **Iter.** gives the number of PRSM iterations, divided by 10^2. For each instance and monomial basis, we run 200 iterations. Then, if the observed value of `UB` satisfies $\mathtt{UB} \leq \mathtt{LB} + 1.5$, we perform 200 additional iterations. We stop early if $\lfloor \mathtt{UB} \rfloor = \mathtt{LB}$. The results show the strength of the SOS_p basis for instances with with 70 variables. In particular, that basis is sufficiently large for closing gaps of several instances. The results also show that the $SOS_p^{\mathbf{Q}}$ basis can be used to further improve bounds for instances with 70, 90 and 110 variables.

In Table 2, we present more details on the best upper bounds attained on the same instances as in Table 1. The columns of Table 2 follow the same definitions as the previous table. Additionally, column $\mathbf{Q}$ relates to the $SOS_p^{\mathbf{Q}}$ basis used and column $|\mathbf{x}|$ reports the number of monomials in that basis (equivalently, the order of the matrix variable of programme $\mathtt{P}_\phi$). Here, we refer to SOS_p as SOS_p^0. Column **T. (s)** reports the computation time in seconds. The results show that we closed the optimality gap for all instances with 70 and 90 variables in less than 9 minutes. Table 2 also shows that the computational time increases w.r.t. the size of the basis. Furthermore, Table 2 shows that uniform random MAX-SAT instances on the same number of variables and clauses can differ in difficulty to solve. For example, instances `s3v70c800-1.cnf` and `s3v70c800-3.cnf` have the same number of variables and clauses. However, proving optimality of the lower bound of the former requires almost three times the computation time as for the latter.

In Figure 1, we show the performance of SOS-MS, using the SOS_p^{50} basis, on all the MAX-3-SAT instances on 70 variables from the MSE-2016. For detailed running times see Table 7 in Appendix B. We compare the running times of SOS-MS with the corresponding running times of the best results of the MSE-2016. That is, for each instance, we compare SOS-MS with the participant of the MSE-2016 that was able to solve that specific instance in the least time. All solvers at the MSE-2016 were tested on an Intel Xeon E5-2620 processor with 2.0 GHz and 3.5 GB RAM[6]. It is hard to compare the running times of algorithms on different machines, since it depends on many factors such as the processor, RAM, the operating system, et cetera. We choose to consider the difference in clock speeds, i.e., 2.8 GHz vs. 2.0 GHz, as well as wall time. Therefore, in Figure 1 and Table 7, we have multiplied all the original running times of SOS-MS with 1.4. Additionally, as the solvers in the MSE-2016 were given a maximum time of 30 minutes per instance, we provided SOS-MS with a maximum of $30/1.4 (\approx 21.4)$ minutes.

Under these constraints, SOS-MS is able to solve all 45 instances in category MAX-3-SAT on 70 variables. The participants from the MSE-2016 could solve at most 42 instances. Specifically, they were unable to solve `s3v70c1500-1.cnf`, `s3v70c1500-4.cnf` and `s3v70c1500-5.cnf`. For instances with a lower number of clauses (around 800) however, the best times per instance of the MSE-2016 are much lower than for SOS-MS. SOS-MS takes on average 309.10 seconds per solved instance, compared to 367.7 seconds per solved instance for the best MSE-2016 solvers. We have also tested SOS-MS using the SOS_p basis, on the same instances. With this different basis, SOS-MS was also able to solve all 45 instances, taking on average 316.7 seconds per instance.

We also investigate the performance of SOS-MS on 80 variable MAX-3-SAT. We consider instances from the MSE-2016 database, although the MSE-2016 did not test 80 variable MAX-3-SAT, and so, it is not known what the best solver per instance. However, we compare SOS-MS to the CCLS2akms MAX-SAT algorithm. This algorithm first runs CCLS [35] to find a good starting lower bound for the

[6]The full specifications of this machine are available at `http://maxsat.ia.udl.cat/machinespecifications/`.

MAX-SAT solution. It then passes this lower bound to the akmaxsat algorithm [29], which solves the instance to optimality. Out of all the publicly available solvers, CCLS2akms[7] placed highest in the random MAX-SAT category of MSE-2016[8].

Results for the 80 variable MAX-3-SAT instances from MSE-2016 are given in Figure 2. The running times per tested MAX-3-SAT instance are provided in Appendix B, Table 8. Both SOS-MS and CCLS2akms are provided a maximum of 30 minutes per instance and are tested on the same hardware (our 16 GB RAM laptop), and thus not scaled. For SOS-MS, we used the SOS_p basis, as $SOS_p^{\mathbf{Q}}$ for $\mathbf{Q} = 40$ and $\mathbf{Q} = 55$ provided worse results. CCLS2akms is able to solve 40 of the 48 instances within the time limit, while SOS-MS solves 43. SOS-MS is however slower: it requires on average 772.53 seconds per solved instance, compared to 370.48 per instance. Again, SOS-MS performs well for instances with a large number of clauses, but requires more time for those with a low number of clauses.

10.2 Weighted partial MAX-2-SAT instances

We show the performance of our solver on (weighted) partial MAX-2-SAT instances from MSE-2016. For this purpose, we adjust SOS-MS by using the theory outlined in Section 8.

In particular, we perform a B&B search, using (63) to compute upper bounds. However, we first preprocess an instance by using the SAT resolution rule (66) on the hard clauses until we find all implied hard clauses of length two or less. Note that this preprocessing may result in improved upper bounds, as described in Section 8.2. If hard unit clauses are found this way, we perform unit resolution and continue with the reduced problem. As initial lower bound for our solver, we take the best known lower bound reported in MSE-2016.

Note that in case of a (weighted) partial MAX-SAT instance, truth values assigned during branching might create hard unit clauses, which leads to more forced truth assignments. Additionally, if a node contains few unassigned variables, determining good upper bounds can be done with a small monomial basis, such as SOS_s^{θ} (31) for small values of θ. Let us describe the choice of θ through the B&B tree. We initialize $\theta_{\text{start}} = 0$. At a node in which we compute bounds, we compute an upper bound $\texttt{UB}$ to the (weighted) partial MAX-SAT solution by first using the basis $SOS_s^{\theta_{\text{start}}}$. If $\lfloor \texttt{UB} \rfloor \leq \texttt{LB}$, we prune the current node. If not, we consider the value $\texttt{GAP} = \texttt{UB} - \texttt{LB} > 0$. When $\texttt{GAP} < 10$, we set $\theta = 0.5$ and recompute a stronger upper bound. For $\texttt{GAP} \geq 10$, we recompute an upper bound with $\theta = 1$ instead. In both cases, we use the PRSM variables corresponding to θ_{start} as warm starts for the next PRSM. If the upper bound obtained using $\theta \in \{0.5, 1\}$ is not equal $\texttt{LB}$, we set $\theta_{\text{start}} = 0.1$ for the remainder of the algorithm. In Appendix C we demonstrate our branching rule and basis selection on illustrative examples, see Figures 3 to 5.

In case the tighter upper bound (corresponding to either $\theta = 0.5$ or $\theta = 1$) does not equal $\texttt{LB}$, we branch at this node. We determine our branching variable in the following way. For n' ($n' \leq n$) the number of unassigned variables at the current node, we consider the five truth assignments that create the largest number of hard unit clauses.

For each of these five truth assignments σ_i, $i \in [5]$, $\sigma_i \in \{-n', \ldots, n'\} \backslash \{0\}$, we compute $\phi_i = \texttt{unitRes}(\phi, \sigma_i)$, see Section 9. Lastly, we select the truth assignment σ_i for which the polynomial ϕ_i has the highest constant term, see (27). The branching variable is then given by $|\sigma_i|$. This branching rule aims to create nodes with many assigned variables, due to the hard unit clauses. This allows for setting θ_{start} to small values, while still providing strong bounds, see also Appendix C.

The computation of SOS-SDP based upper bounds is expensive, compared to bounding methods used in other MAX-SAT solvers. Therefore, we only compute bounds at selected nodes (see also [52]). Our selection process is described in detail in Appendix D.

We test the described procedure on the 60 unweighted partial MAX-2-SAT, and the 90 weighted partial MAX-2-SAT instances from the MSE-2016, setting a maximum time of 30 minutes per instance. Each instance contains 150 variables and 150 hard clauses. The number of total clauses (both soft and hard) ranges from 1000 to 5000, and all of them have length two. In the weighted variant, soft clause weights range from 1 up to and including 10. Tables 3 and 4 report the running times per instance (rounded to the nearest second), for unweighted and weighted partial MAX-2-SAT, respectively. Here we provide original runtimes, thus not multiplied by some factor. A '-' value indicates a time-out of 30 minutes. Variable m denotes the number of total clauses. The **Instance** row corresponds to the instance file name, as taken from the MSE-2016. For example,

[7] The CCLS2akms algorithm is available at `http://www.maxsat.udl.cat/16/solvers/index.html`.
[8] The MSE-2016 results are available at `http://maxsat.ia.udl.cat/docs/ms.pdf`.

$m = 2500$ and **Instance** $= 1$ refer to `file_rpms_wcnf_L2_V150_C2500_H150_1.wcnf` in Table 3, and `file_rwpms_wcnf_L2_V150_C2500_H150_1.wcnf` in Table 4.

The table shows that we are able to solve many instances within the 30-minute time limit. This shows the strength of SDP applied also to the (weighted) partial MAX-SAT. Since we are first to solve the (weighted) partial MAX-SAT by using SDP approaches, our work opens new perspectives on solving variants of the MAX-SAT.

10.3 Partial MAX-3-SAT

We show the quality of the SDP bounds for partial MAX-3-SAT, based on 10 instances from the MSE-2016. Each instance contains 500 soft clauses and 100 hard clauses, all of length three.

We compute an upper bound for each instance ϕ (on n variables, having hard clauses $C_p^{\mathbf{H}}$) in the following way. We first perform SAT resolution (66) on the hard clauses to find all implied hard clauses of length 4 or less. Then, for $\mathbf{Q} \in \mathbb{N}$, we consider the $\mathbf{Q}$ variables that appear in the largest number of (soft and hard) clauses. Let $V \subseteq [n]$ be the subset indicating those variables. We construct additional hard clauses of the form $C_p^{\mathbf{H}} \vee x_i$, for all $i \in V$ and $C_p^{\mathbf{H}} \subseteq V$, with $|C_p^{\mathbf{H}}| \leq 3$. Note that these additional hard clauses do not change the set of satisfying assignments. On so generated instance we compute an upper bound using the $SOS_p^{\mathbf{Q}}$ basis, see (30). Note that, as the newly generated clauses $C_p^{\mathbf{H}} \vee x_i$ are contained in V, they create no additional monomials in the $SOS_p^{\mathbf{Q}}$. This ensures that the size of the matrix variable remains manageable. Moreover, these additional hard clauses strengthen the bound, see Section 8.2.

We perform this procedure for each instance and $\mathbf{Q} \in \{70, 75, 80, 85, 90\}$, for a time limit of 30 minutes. These values of $\mathbf{Q}$ are chosen with the goal of computing the tightest bound at the 30-minute mark. We report results in Table 6. Column **Inst.** reports the instance file name, according to the naming scheme `file_rpms_wcnf_L3_V100_C600_H100_`[**Inst.**]`.wcnf`. Each instance has 600 clauses, of which 100 are hard, all of length three, on 100 variables. Column `LB` reports the optimal lower bound, as verified by solvers in the MSE-2016. Column $\mathbf{Q}$ refers to the $SOS_p^{\mathbf{Q}}$ basis used, of which the number of monomials is reported in column $|\mathbf{x}|$. Column $|C^{\mathbf{H}}|$ reports the number of hard clauses used. The next 6 columns (`GAP` at $\mathbf{X}$ minutes) report the value of the `GAP` (i.e., `UB - LB`) at different time points. To compute the values of `UB`, we compute the smallest eigenvalue of M using the LOBPCG algorithm [28], see (55). Lastly, as a measure of convergence, the final column reports the (absolute) value of the smallest eigenvalue of the matrix variable at the final iteration, multiplied by the size of this matrix. This column thus reports the difference in trace between $\widetilde{M}$ and M, see (55).

Considering the bound at 30 minutes, the SOS_p^{85} basis performs best. A larger basis is unable to converge in 30 minutes, while smaller bases result in weaker bounds. Since differences in bounds for different $\mathbf{Q}$ are small, smaller bases might be more useful in combination with a B&B scheme.

10.4 Weighted MAX-3-SAT

Lastly, we test SOS-MS for some weighted MAX-3-SAT instances.

We consider 10 weighted MAX-3-SAT instances from the MSE-2016. Each instance contains 70 variables, and either 1400 or 1500 weighted soft clauses. These weights range between 1 and 10. There are no hard clauses.

For the weighted MAX-3-SAT, we compare the running times of SOS-MS with CCLS2akms, on the same hardware. For SOS-MS, we attempted multiple monomial bases, and found that the SOS_p basis required the least time to solve the weighted MAX-3-SAT instances.

The running times per instance are reported in Table 5. Column m reports the number of clauses, and **Inst.** reports the instance, according to the scheme `s3v70c`[m]`-`[**Inst.**]`.wcnf`. SOS-MS is able to solve three instances in less time than CCLS2akms and can solve 9 of the 10 instances in less than 30 minutes. This demonstrates that SOS-MS is well suited for solving weighted MAX-3-SAT instances with a large number of clauses.

11 Conclusions and future work

In this paper we consider SOS optimization for solving the MAX-SAT and weighted partial MAX-SAT. We design an SOS-SDP based exact MAX-SAT solver, called SOS-MS. Our solver is competitive with the best-known solvers on solving various (weighted partial) MAX-SAT instances. We are also first to compute SDP bounds for the weighted partial MAX-SAT.

In Section 3 we propose a family of semidefinite feasibility problems $R_{\mathcal{F}}(\phi)$ and show that one member of this family provides the rank two guarantee, see Theorem 1. That is, the existence of a feasible rank two matrix implies satisfiability of the corresponding SAT instance. In Section 4, we outline the SOS approach to the MAX-SAT, due to van Maaren et al. [51] and propose new bases. We introduce the SOS_s^θ and $SOS_p^{\mathbf{Q}}$ bases, see (31), and provide several theoretical results related to these bases, see Lemmas 1 and 2. Clearly, the strength of the SOS-SDP based relaxations and the required time to compute them depend on the chosen monomial basis. The SOS-SDP relaxation for the MAX-SAT is denoted by $\mathtt{P}_\phi$. We consider MAX-SAT resolution in Section 5 and show that resolution might not be beneficial for the SOS approach applied to the MAX-SAT.

In Section 6, we elegantly show a connection between the SOS approach to the MAX-SAT and the family of semidefinite feasibility problems $R_{\mathcal{F}}(\phi)$. This is done by deriving the dual problem to $\mathtt{P}_\phi$, see Theorem 2. In Section 7, we propose PRSM for solving $\mathtt{P}_\phi$. We show that PRSM is well suited for exploiting the structure of $\mathtt{P}_\phi$, in particular, the unit constraints, see (35). We thus provide an affirmative answer to the key question posed by van Maaren et al. [51]: *"Whether SDP software can be developed dealing with unit constraints efficiently?"*.

We extend the SOS approach for the MAX-SAT to the weighted partial MAX-SAT in Section 8. Here, the variables are restricted to satisfy a set of hard clauses. We show that such hard clauses can be incorporated in the SOS programme $\mathtt{P}_\phi$ by adding scalar variables. We show in Section 8.1 that the resulting programme (62) is also well suited for the PRSM.

In Section 9, we provide implementation details of our SOS-SDP based MAX-SAT solver, whose pseudocode is given in Algorithm 1. SOS-MS is a B&B algorithm and has two crucial components. The first one is the use of warm starts to programme $\mathtt{P}_\phi$, in order to quickly obtain strong bounds. The second one is its ability to quickly parse $\mathtt{P}_\phi$, as outlined in Section 9.1. Our algorithm parses a basis that contains 4,439,449 monomials in (approximately) one second (!).

In Section 10 we provide extensive numerical results that verify efficiency of our exact solver SOS-MS and quality of SOS upper bounds. We show that SOS-MS can solve a variety of MAX-SAT instances in reasonable time, while solving some instances faster than the best solvers in the MSE-2016. We show that the $SOS_p^{\mathbf{Q}}$ bases (30) are able to prove optimality of some MAX-SAT instances, and that the parameter $\mathbf{Q}$ provides the option to adjust the trade-off between quality of the bounds and computation time. We also test our B&B algorithm for (weighted) partial MAX-SAT instances in Sections 10.2 and 10.4. Our solver is able to solve many (weighted) partial MAX-SAT instances in a reasonable time.

This paper has demonstrated the strong performance of SOS-MS on (weighted partial) MAX-SAT instances from the MSE random track. In the future, we hope to also solve instances with SOS-MS from the so-called *industrial* and *crafted* tracks. These tracks currently impose two challenges on SOS-MS. Firstly, these instances induce prohibitively large SOS_p bases, which hinders the computation of strong bounds. To solve this, we require a more sophisticated method for choosing a smaller, manageable, basis, like SOS_s^θ. Secondly, these instances can possess clauses of length k, where $k \geq 4$. This is problematic in the current settings, since F_ϕ, see (21), is a kth degree polynomial, which requires a large basis to be represented. One possible way to overcome these challenges is through exploiting the structure present in these instances. For example, function F_ϕ might have few nonzero coefficients, which allows for finding SOS decompositions with small monomial bases, see also [2].

		SOS_p		SOS_p^{40}		SOS_p^{50}		SOS_p^{60}		SOS_p^{70}		SOS_p^{110}	
Instance	LB	UB	**Iter.**	UB	**Iter.**	UB	**Iter.**	UB	**Iter.**	UB	**Iter.**	UB	**Iter.**
s3v70c800-1	769	771.29	2	770.79	2	770.67	2	769.99	3.9				
s3v70c800-3	770	770.996	3										
s3v70c800-4	772	772.99	2.8										
s3v70c900-4	861	863.11	2	862.63	2	861.99	3.3						
s3v70c1000-1	953	954.89	2	954.70	2	954.65	2	953.999	3.5				
s3v70c1000-2	957	957.99	2.7										
s3v70c1000-5	958	958.996	2.2										
s3v70c1100-4	1048	1049.03	4	1048.997	3.2								
s3v70c1500-2	1411	1411.998	2.8										
s3v90c900-5	875	877.44	2	875.99	2.9								
s3v90c900-7	873	877.16	2	875.56	2	875.05	2	874.62	2	873.995	2.7		
s3v110c1000-7	969	984.01	2	980.93	2	979.77	2	978.63	2	977.61	2	974.49	2
s3v110c1100-10	1064	1076.81	2	1074.15	2	1073.03	2	1071.98	2	1071.01	2	1068.32	2

Table 1: Comparison of the MAX-3-SAT bounds attained by different monomial bases.

Instance	LB	UB	T. (s)	Q	|x|	Iter.
s3v70c800-1	769	769.99	243.0	60	2181	3.9
s3v70c800-3	770	770.996	85.0	0	1603	3.0
s3v70c800-4	772	772.99	80.7	0	1588	2.8
s3v70c900-4	861	861.99	168.4	50	2022	3.3
s3v70c1000-1	953	953.999	236.6	60	2244	3.5
s3v70c1000-2	957	957.99	102.4	0	1810	2.7
s3v70c1000-5	958	958.996	83.6	0	1798	2.2
s3v70c1100-4	1048	1048.997	164.4	40	2014	3.2
s3v70c1500-2	1411	1411.998	165.7	0	2130	2.8
s3v90c900-5	875	875.99	218.4	40	2366	2.9
s3v90c900-7	873	873.995	519.9	70	3185	2.7
s3v110c1000-7	969	974.49	3089.3	110	6106	2.0
s3v110c1100-10	1064	1068.32	3232.5	110	6106	2.0

Table 2: Best upper bounds for the MAX-3-SAT per instance

Figure 1: SOS-MS on 70 variable MAX-3-SAT (basis SOS_p^{50})

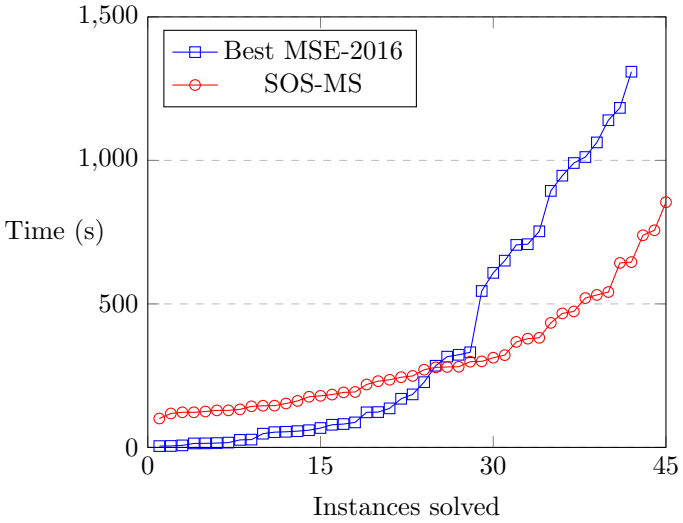
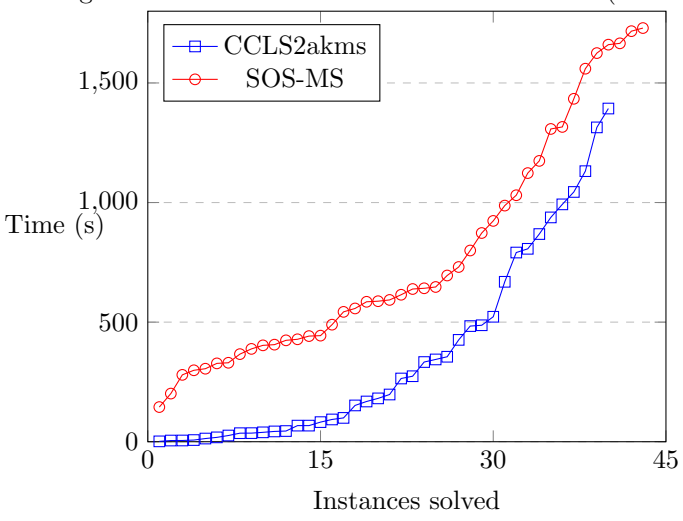


Figure 2: SOS-MS on 80 variable MAX-3-SAT (basis SOS_p)



		Instance									
		0	1	2	3	4	5	6	7	8	9
m	2500	59	177	237	536	75	45	400	361	320	61
	3000	327	329	96	244	103	554	675	86	339	221
	3500	322	223	112	134	188	620	23	469	144	252
	4000	85	5	107	127	347	662	762	320	-	289
	4500	679	334	592	251	732	470	145	223	318	135
	5000	159	116	663	1258	975	226	473	598	200	105

Table 3: Unweighted partial MAX-2-SAT running times (seconds)

		Instance									
		0	1	2	3	4	5	6	7	8	9
m	1000	116	164	61	40	54	85	55	196	344	68
	1500	608	144	447	164	529	190	105	269	349	370
	2000	325	495	326	222	134	233	124	156	178	631
	2500	103	544	282	1029	318	315	575	926	619	118
	3000	-	1341	446	-	249	-	1220	422	1624	618
	3500	1667	1195	1022	450	1327	1351	130	771	196	229
	4000	91	5	208	1108	930	-	-	1048	-	338
	4500	-	-	1601	1220	-	732	518	1347	799	965
	5000	338	980	-	686	-	-	-	-	222	294

Table 4: Weighted partial MAX-2-SAT running times (seconds)

References

[1] A. Abramé and D. Habet. AHMAXSAT: Description and evaluation of a branch and bound Max-SAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 9(1):89–128, 2014.

[2] A. A. Ahmadi, G. Hall, A. Papachristodoulou, J. Saunderson, and Y. Zheng. Improving efficiency and scalability of sum of squares optimization: Recent advances and limitations. In *2017 IEEE 56th annual conference on decision and control (CDC)*, pages 453–462. IEEE, 2017.

[3] M. F. Anjos. Proofs of unsatisfiability via semidefinite programming. In *Operations Research Proceedings 2003*, pages 308–315. Springer, 2004.

[4] M. F. Anjos. On semidefinite programming relaxations for the satisfiability problem. *Mathematical Methods of Operations Research*, 60(3):349–367, 2004.

[5] M. F. Anjos. An improved semidefinite programming relaxation for the satisfiability problem. *Mathematical Programming*, 102(3):589–608, 2005.

		Running time (s)	
m	**Inst.**	**SOS-MS**	**CCLS2akms**
1400	1	438.71	322.62
	2	795.71	278.78
	3	696.16	752.95
	4	852.51	592.82
	5	1250.02	1054.16
1500	1	499.26	885.66
	2	653.51	1011.90
	3	399.28	292.60
	4	791.08	748.03
	5	> 1800	1323.78

Table 5: Weighted 70 variable MAX-3-SAT

Inst.	LB	Q	$\|\mathbf{x}\|$	$\|C^{\mathbf{H}}\|$	GAP at **X** minutes						$\lambda_{\min} \cdot \|\mathbf{x}\|(\times 10^{-2})$
					5	10	15	20	25	30	
0	492	70	3288	5188	4.65	3.55	3.16	2.95	2.82	2.72	4.01
		75	3507	6124	4.81	3.51	3.07	2.82	2.67	2.56	5.82
		80	3754	7140	5.35	3.61	3.03	2.72	2.53	2.40	9.64
		85	4035	7908	6.31	4.08	3.20	2.78	2.52	2.34	18.54
		90	4340	8629	7.54	4.76	3.70	3.14	2.78	2.53	41.27
1	492	70	3334	4412	4.37	3.48	3.16	2.97	2.84	2.75	2.80
		75	3548	5509	5.03	3.58	3.13	2.88	2.72	2.60	4.49
		80	3780	6331	5.56	3.66	3.11	2.81	2.63	2.50	6.95
		85	4054	7377	5.84	3.60	2.97	2.65	2.44	2.30	9.41
		90	4352	8329	7.77	4.44	3.46	2.97	2.67	2.46	25.62
2	492	70	3324	5067	4.09	2.99	2.58	2.35	2.20	2.10	5.29
		75	3536	5993	4.47	3.05	2.56	2.29	2.11	1.98	9.78
		80	3776	7076	4.92	3.18	2.60	2.27	2.06	1.91	15.80
		85	4046	7839	5.05	3.25	2.51	2.16	1.93	1.77	23.72
		90	4346	8729	7.02	4.12	3.18	2.67	2.34	2.10	52.53
3	491	70	3296	4964	5.33	4.42	4.04	3.82	3.68	3.58	1.58
		75	3528	5741	5.33	4.26	3.84	3.60	3.44	3.33	2.24
		80	3773	6501	5.79	4.26	3.76	3.49	3.31	3.18	3.47
		85	4038	7475	5.94	4.15	3.62	3.33	3.14	3.00	4.56
		90	4343	8345	8.18	4.92	4.02	3.58	3.31	3.12	10.29
4	493	70	3304	4832	4.10	3.14	2.74	2.52	2.38	2.28	3.24
		75	3529	5815	4.54	3.17	2.70	2.44	2.26	2.13	5.41
		80	3775	6811	4.62	3.06	2.55	2.27	2.08	1.95	7.06
		85	4052	7641	5.23	3.17	2.55	2.21	1.99	1.84	11.19
		90	4352	8695	7.53	4.16	3.11	2.60	2.28	2.05	25.95
5	492	70	3287	5207	3.96	3.15	2.82	2.64	2.52	2.44	1.80
		75	3506	6071	4.06	3.15	2.74	2.52	2.37	2.27	2.82
		80	3760	6851	4.40	3.11	2.66	2.40	2.24	2.12	4.29
		85	4037	7927	4.99	3.29	2.76	2.46	2.26	2.12	7.77
		90	4347	8648	6.70	4.05	3.25	2.83	2.54	2.34	22.01
6	493	70	3298	4633	3.70	2.84	2.49	2.29	2.16	2.07	3.60
		75	3508	5745	3.94	2.82	2.42	2.19	2.03	1.93	5.45
		80	3744	6659	4.27	2.82	2.38	2.13	1.97	1.84	9.95
		85	4018	7643	4.87	3.11	2.38	2.06	1.86	1.71	15.91
		90	4332	8523	7.29	3.97	3.03	2.55	2.25	2.02	40.32
7	492	70	3345	4980	5.38	4.44	4.05	3.83	3.68	3.58	1.26
		75	3562	6117	5.48	4.42	3.97	3.72	3.55	3.43	1.75
		80	3803	7056	5.89	4.42	3.92	3.62	3.43	3.30	2.50
		85	4062	7901	5.88	4.25	3.73	3.43	3.23	3.09	3.02
		90	4359	8709	7.50	4.68	3.93	3.56	3.31	3.12	8.27
8	495	70	3258	5155	3.83	2.65	2.23	1.99	1.84	1.72	9.72
		75	3484	5803	4.09	2.65	2.17	1.90	1.72	1.59	15.76
		80	3735	6722	4.85	3.02	2.29	1.96	1.75	1.59	27.59
		85	4008	7547	5.11	3.08	2.37	1.90	1.66	1.50	39.49
		90	4324	8770	7.55	4.18	3.09	2.52	2.17	1.92	76.63
9	491	70	3341	5206	5.33	4.39	4.00	3.78	3.63	3.53	1.93
		75	3569	5926	5.43	4.31	3.85	3.59	3.42	3.31	2.59
		80	3811	7158	5.78	4.27	3.75	3.46	3.27	3.14	3.60
		85	4078	8254	5.92	4.17	3.63	3.32	3.11	2.97	4.55
		90	4367	9169	7.57	4.80	4.00	3.56	3.28	3.08	11.35

Table 6: Bounds for partial MAX-3-SAT instances

[6] M. F. Anjos. Semidefinite optimization approaches for satisfiability and maximum-satisfiability problems. *Journal on Satisfiability, Boolean Modeling and Computation*, 1(1):1–47, 2006.

[7] M. F. Anjos. An extended semidefinite relaxation for satisfiability. *Journal on Satisfiability, Boolean Modeling and Computation*, 4(1):15–31, 2007.

[8] M. F. Anjos and H. Wolkowicz. Geometry of semidefinite Max-Cut relaxations via matrix ranks. *Journal of Combinatorial Optimization*, 6(3):237–270, 2002.

[9] R. Asín Achá and R. Nieuwenhuis. Curriculum-based course timetabling with SAT and MaxSAT. *Annals of Operations Research*, 218(1):71–91, 2014.

[10] M. L. Bonet, J. Levy, and F. Manya. Resolution for Max-SAT. *Artificial Intelligence*, 171(8-9): 606–618, 2007.

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[12] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[13] E. de Klerk, H. van Maaren, and J. Warners. Relaxations of the satisfiability problem using semidefinite programming. *Journal of automated reasoning*, 24(1):37–65, 2000.

[14] F. de Meijer and R. Sotirov. SDP-based bounds for the quadratic cycle cover problem via cutting-plane augmented Lagrangian methods and reinforcement learning: Informs journal on computing meritorious paper awardee. *INFORMS Journal on Computing*, 33(4):1262–1276, 2021.

[15] D. Drusvyatskiy, G. Li, and H. Wolkowicz. A note on alternating projections for ill-posed semidefinite feasibility problems. *Mathematical Programming*, 162(1):537–548, 2017.

[16] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximations. *Computers and Mathematics with Applications*, 2(7):17—40, 1976.

[17] P. Gattermann, P. Großmann, K. Nachtigall, and A. Schöbel. Integrating passengers' routes in periodic timetabling: a SAT approach. In *16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

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

[19] N. Graham, H. Hu, J. Im, X. Li, and H. Wolkowicz. A restricted dual Peaceman-Rachford splitting method for a strengthened DNN relaxation for QAP. *INFORMS Journal on Computing*, 2022.

[20] E. Halperin and U. Zwick. Approximation algorithms for MAX 4-SAT and rounding procedures for semidefinite programs. *Journal of Algorithms*, 40(2):184–211, 2001.

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

[22] B. He, F. Ma, and X. Yuan. Convergence study on the symmetric version of ADMM with larger step sizes. *SIAM Journal on Imaging Sciences*, 9(3):1467–1501, 2016.

[23] D. Henrion and J. Malick. Projection methods for conic feasibility problems: applications to polynomial sum-of-squares decompositions. *Optimization Methods & Software*, 26(1):23–46, 2011.

[24] D. Henrion, J. B. Lasserre, and J. Löfberg. GloptiPoly 3: moments, optimization and semidefinite programming. *Optimization Methods & Software*, 24(4-5):761–779, 2009.

[25] D. Jagt, S. Shivakumar, P. Seiler, and M. Peet. Efficient data structures for exploiting sparsity and structure in representation of polynomial optimization problems: Implementation in SOSTOOLS. *arXiv preprint arXiv:2203.01910*, 2022.

[26] H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 406–415. IEEE, 1997.

[27] H. Kautz and B. Selman. Unifying SAT-based and graph-based planning. In *IJCAI*, volume 99, pages 318–325, 1999.

[28] A. V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing*, 23(2):517–541, 2001.

[29] A. Kuegel. Improved exact solver for the weighted MAX-SAT problem. *EPiC Series in Computing*, 8:15–27, 2012.

[30] J. B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001.

[31] J. B. Lasserre. A sum of squares approximation of nonnegative polynomials. *SIAM review*, 49(4): 651–669, 2007.

[32] M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, pages 157–270. Springer, 2009.

[33] M. Lewin, D. Livnat, and U. Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 67–82. Springer, 2002.

[34] C. M. Li and F. Manya. MaxSAT, hard and soft constraints. In *Handbook of satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications,* Editors, Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh, pages 903–927. IOS Press, 2021.

[35] C. Luo, S. Cai, W. Wu, Z. Jie, and K. Su. CCLS: An efficient local search algorithm for weighted maximum satisfiability. *IEEE Transactions on Computers*, 64(7):1830–1843, 2015.

[36] J. P. Marques-Silva and K. A. Sakallah. Boolean satisfiability in electronic design automation. In *Proceedings of the 37th Annual Design Automation Conference*, pages 675–680, 2000.

[37] M. Mendonca, A. Wąsowski, and K. Czarnecki. SAT-based analysis of feature models is easy. In *Proceedings of the 13th International Software Product Line Conference*, pages 231–240, 2009.

[38] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019. URL `http://docs.mosek.com/9.0/toolbox/index.html`.

[39] D. E. Oliveira, H. Wolkowicz, and Y. Xu. ADMM for the SDP relaxation of the QAP. *Mathematical Programming Computation*, 10(4):631–658, 2018.

[40] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, P. A. Parrillo, M. M. Peet, and D. Jagt. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB.* `http://arxiv.org/abs/1310.4716`, 2021. Available from `https://github.com/oxfordcontrol/SOSTOOLS`.

[41] P. A. Parrilo and R. R. Thomas, editors. *Sum of Squares: Theory and Applications*, volume 77 of *Proceedings of Symposia in Applied Mathematics*. American Mathematical Society, Providence, Rhode Island, 2020.

[42] D. W. Peaceman and H. H. Rachford, Jr. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for industrial and Applied Mathematics*, 3(1):28–41, 1955.

[43] F. Permenter and P. A. Parrilo. Basis selection for SOS programs via facial reduction and polyhedral approximations. In *53rd IEEE Conference on Decision and Control*, pages 6615–6620. IEEE, 2014.

[44] M. R. Prasad, A. Biere, and A. Gupta. A survey of recent advances in SAT-based formal verification. *International Journal on Software Tools for Technology Transfer*, 7(2):156–173, 2005.

[45] M. Putinar. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal*, 42(3):969–984, 1993.

[46] B. Reznick. Extremal PSD forms with few terms. *Duke mathematical journal*, 45(2):363–374, 1978.

[47] N. Rontsis, P. Goulart, and Y. Nakatsukasa. Efficient semidefinite programming with approximate ADMM. *Journal of Optimization Theory and Applications*, 192(1):292–320, 2022.

[48] C. Scheiderer. Positivity and sums of squares: a guide to recent results. In *Emerging applications of algebraic geometry*, pages 271–324. Springer, 2009.

[49] B. Sturmfels. Polynomial equations and convex polytopes. *The American mathematical monthly*, 105(10):907–922, 1998.

[50] H. van Maaren and L. van Norden. Sums of squares, satisfiability and maximum satisfiability. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 294–308. Springer, 2005.

[51] H. van Maaren, L. van Norden, and M. J. Heule. Sums of squares based approximation algorithms for MAX-SAT. *Discrete Applied Mathematics*, 156(10):1754–1779, 2008.

[52] P.-W. Wang and J. Zico Kolter. Low-rank semidefinite programming for the MAX2SAT problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1641–1649, 2019.

[53] Y. Zheng, G. Fantuzzi, and A. Papachristodoulou. Exploiting sparsity in the coefficient matching conditions in sum-of-squares programming using ADMM. *IEEE control systems letters*, 1(1):80–85, 2017.

A PRSM implementation detail

For the PRSM scheme (49), one only requires access to the matrix $\frac{1}{\beta}S^k$, instead of S^k. We therefore propose the following simple adaption to the PRSM scheme, which maintains the loop invariant $Q^k = \frac{1}{\beta}S^k$.

$$\begin{cases} Z^{k+1} &= \mathcal{P}_{\mathcal{S}_+}\left(M^k + Q^k\right), \\ Q^{k+1/2} &= Q^k + \gamma_1(M^k - Z^{k+1}), \\ M^{k+1} &= \mathcal{P}_{\mathcal{M}_\phi}\left(Z^{k+1} - \frac{1}{\beta}I - Q^{k+1/2}\right), \\ Q^{k+1} &= Q^{k+1/2} + \gamma_2(M^{k+1} - Z^{k+1}). \end{cases}$$

Compared to (49), the above scheme does not require the computation of $\frac{1}{\beta}S^k$ twice per iteration.

B Runtimes per MAX-3-SAT instance

We provide the running times of SOS-MS and the best performing MAX-SAT algorithms from MSE-2016 per tested instance. Table 7 reports running times per tested 70 variable MAX-3-SAT instance, corresponding to Figure 1. Column m indicates the number of clauses. Column **Inst.** reports the name of the instance (e.g., $m = 900$ and **Inst.**$= 3$ corresponds to `s3v70c900-3.cnf`). The original runtimes of SOS-MS are here multiplied by a factor 1.4, to account for the different machine compared to those in MSE-2016. A '-' value in Table 7 indicates a time-out (of 30 minutes for MSE-2016 and 30/1.4 minutes for SOS-MS). The instances `s3v70c1500-1.cnf`, `s3v70c1500-4.cnf` and `s3v70c1500-5.cnf` remained unsolved in the MSE-2016. Using SOS-MS, we compute that their optimal values are 1410, 1409 and 1406, respectively.

Table 8 reports running times per tested 80 variable MAX-3-SAT instance, corresponding to Figure 2. Here, column m indicates the number of clauses per instance, **Inst.** column reports the name of the instance (e.g., $m = 1000$ and **Inst.**$= 4$ corresponds to `s3v80c1000-4.cnf`). Columns **SOS-MS** and **CCLS2akms** provide the running times of each solver, in seconds. Here we provide original computational times, thus not scaled ones.

m	Inst.	Running time (s) SOS-MS	MSE
700	1	179.74	5.00
	2	145.49	7.06
	3	298.63	14.14
	4	381.82	13.83
	5	145.19	4.61
800	1	281.03	27.20
	2	378.84	81.33
	3	125.14	16.76
	4	129.03	15.14
	5	118.23	26.16
900	1	280.31	67.52
	2	235.61	59.97
	3	132.43	53.09
	4	219.21	54.46
	5	321.44	87.16
1000	1	312.57	123.08
	2	193.14	56.66
	3	122.33	78.94
	4	100.70	122.22
	5	143.14	47.84
1100	1	531.20	331.47
	2	244.50	227.34
	3	122.36	136.06
	4	230.81	169.47
	5	176.46	184.75
1200	1	756.47	752.93
	2	161.94	322.68
	3	645.56	608.15
	4	739.14	1011.74
	5	473.66	544.98
1300	1	520.31	1139.98
	2	541.27	946.71
	3	367.58	708.07
	4	153.01	316.19
	5	278.93	650.70
1400	1	129.06	705.64
	2	854.36	1309.01
	3	249.01	990.89
	4	184.05	284.52
	5	299.69	1062.63
1500	1	466.85	-
	2	270.72	1182.99
	3	191.68	893.82
	4	642.74	-
	5	433.93	-

Table 7: 70 variable MAX-3-SAT instances

m	Inst.	Running time (s) SOS-MS	CCLS2akms
700	1	441.66	6.98
	2	638.47	13.29
	3	489.35	4.95
	4	144.89	1.52
	5	799.49	18.21
	6	402.94	5.16
800	1	923.21	44.68
	2	-	197.40
	3	1174.15	43.04
	4	557.18	25.93
	5	388.10	35.73
	6	1123.17	39.26
900	1	585.08	81.99
	2	-	181.21
	3	730.48	99.57
	4	305.24	67.33
	5	423.89	67.35
	6	298.67	35.94
1000	1	1316.13	355.26
	2	1307.67	273.86
	3	327.52	93.09
	4	1559.63	483.51
	5	405.84	151.73
	6	366.26	168.22
1100	1	1624.88	937.81
	2	-	1393.08
	3	279.54	344.01
	4	1716.55	806.55
	5	587.51	264.34
	6	1660.14	425.80
1200	1	542.94	790.99
	2	201.26	333.18
	3	592.83	522.42
	4	641.53	486.97
	5	1030.56	868.28
	6	1729.55	1131.22
1300	1	330.12	668.63
	2	614.47	1314.20
	3	-	-
	4	428.29	992.42
	5	-	-
	6	1665.65	-
1400	1	443.53	-
	2	872.60	-
	3	695.00	-
	4	646.95	1044.26
	5	987.23	-
	6	1433.79	-

Table 8: 80 variable MAX-3-SAT instances

C Search tree for the partial MAX-2-SAT

We provide the search trees of our SOS-SDP based algorithm for solving various partial MAX-2-SAT instances, as described in Section 10.2. These instances are also reported in Tables 3 and 4.

For the search trees in Figures 3 to 5, each node is given a numeric value between zero and one, or the value **B**. Numeric values indicate the largest value of θ for which basis SOS_s^θ was used to compute an upper bound in that node. The value **B** (**B** for branch) indicates that no upper bound was computed in this node, but instead immediately a variable was chosen to branch.

The figures show the strength of the SDP bounds, implying that many nodes can be pruned immediately. This also demonstrates the effectiveness of the branching rule, which is able to find many nodes that can be pruned immediately.

D Branching process for the partial MAX-2-SAT

During the B&B search for the optimal solution to the partial MAX-2-SAT, we do not compute SOS-SDP based upper bounds at each node. We describe here the process which decides in which nodes the algorithm computes an upper bound.

Recall that our branching rule, described Section 10.2, selects the variable $i \in [n]$ for which either $\texttt{unitRes}(\phi, i)$ or $\texttt{unitRes}(\phi, -i)$ contains many hard unit clauses, and therefore many truth assignments. Each branching step creates two child nodes. We refer to the node which corresponds to the proposition with most truth assignments in the two child nodes as a *bad* node. The forced assignments resulting from the hard unit clauses in that proposition are often sub optimal, which explains the name.

The algorithm for the B&B search operates in two phases, named phase I and phase II. In phase I, we only compute upper bounds for bad nodes. We exit phase I when the algorithm fails to prune a bad node, or when the number of remaining variables is smaller than some fixed value $n_{\min}$. This process is given in pseudocode in Algorithm 2.

After exiting phase I, the algorithm enter phase II, see Algorithm 3. In phase II, before we compute an upper bound in a node, we first attempt to remove variables from the proposition, by pruning bad nodes. This is described in Lines 6 to 12. The main difference with phase I is that, when we fail to prune a bad node in these lines, we do not recompute a stronger upper bound with a larger monomial basis. The extra effort in phase I is justified since pruning a node in phase I equates to removing one unassigned variable from the rest of the search tree.

After Lines 6 to 12, we consider the remaining proposition ϕ, and compute the basis SOS_s^1. If this basis is too large, we branch immediately. Otherwise, we compute an upper bound. We set the parameters as $\theta_{\text{start}} = 0$ in phase I, $\theta_{\text{start}} = 0.1$ in phase II, and $(b_{\max}, n_{\min}, s_{\max}) = (15, 70, 1750)$.

Algorithm 2: B&B search for the (weighted) partial MAX-2-SAT, phase I

Input: Lower bound LB, proposition ϕ, parameters $(\theta_{\text{start}}, n_{\min}) \in [0, 0.5) \times \mathbb{N}$.
Set $\theta = \theta_{\text{start}}$.
while $n_\phi \geq n_{\min}$ **do**
    Determine the truth assignment σ according to the branching rule from Section 10.2.
    Compute $\phi' = \texttt{unitRes}(\phi, \sigma)$.
    /* Proposition ϕ' corresponds to a bad node. */
    Solve $\texttt{P}_{\phi'}$, using basis SOS_s^θ, to obtain UB.
    if $\lfloor \texttt{UB} \rfloor \leq \texttt{LB}$ **then**
        Update $\phi := \texttt{unitRes}(\phi, -\sigma)$, and reset θ by $\theta := \theta_{\text{start}}$.
        /* Note that we did not compute an upper bound for the old ϕ. */
    else
        if $\theta = \theta_{\text{start}}$ **then**
            Set $\theta = 0.5$ if $\texttt{UB} - \texttt{LB} < 10$, set $\theta = 1$ otherwise.
            /* Recompute a stronger upper bound with a larger θ. */
        else
            /* Stronger upper bound was unable to prune the node. */
            Compute $\phi'' = \texttt{unitRes}(\phi, -\sigma)$.
            Add two nodes corresponding to ϕ' and ϕ'' to the search tree.
            break
            /* Move to phase II of the algorithm (see Algorithm 3). */

Algorithm 3: B&B search for the (weighted) partial MAX-2-SAT, phase II

Input: Lower bound LB, parameters $(\theta_{\text{start}}, b_{\max}, n_{\min}, s_{\max}) \in [0, 1) \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$.
while The search tree contains a node which is neither branched nor pruned **do**
    Consider an unbranched and unpruned node in the search tree, with proposition ϕ.
    Set $b = 0$.
    while $b < b_{\max}$ & $n_\phi > n_{\min}$ **do**
        Determine the truth assignment σ according to the branching rule from Section 10.2.
        Compute $\phi' = \texttt{unitRes}(\phi, \sigma)$.
        /* Proposition ϕ' corresponds to a bad node. */
        Solve $\texttt{P}_{\phi'}$, using basis $SOS_s^{\theta_{\text{start}}}$, to obtain UB.
        if $\lfloor \texttt{UB} \rfloor \leq \texttt{LB}$ **then**
            Update $\phi := \texttt{unitRes}(\phi, -\sigma)$, and $b := b + 1$.
            /* Note that we did not compute an upper bound for the old ϕ. */
        else
            break
    Compute monomial basis SOS_s^1 for ϕ.
    if $|SOS_s^1| > s_{\max}$ **then**
        Branch the node corresponding to ϕ, add its two children to the search tree and continue with B&B search.
        /* For efficiency reasons, compute upper bounds only when the basis is small enough. */
    else
        Solve $\texttt{P}_\phi$, using basis SOS_s^1, to obtain UB.
        if $\lfloor \texttt{UB} \rfloor \leq \texttt{LB}$ **then**
            Prune the node corresponding to ϕ, and continue with the B&B search.
        else
            Perform Line 15.

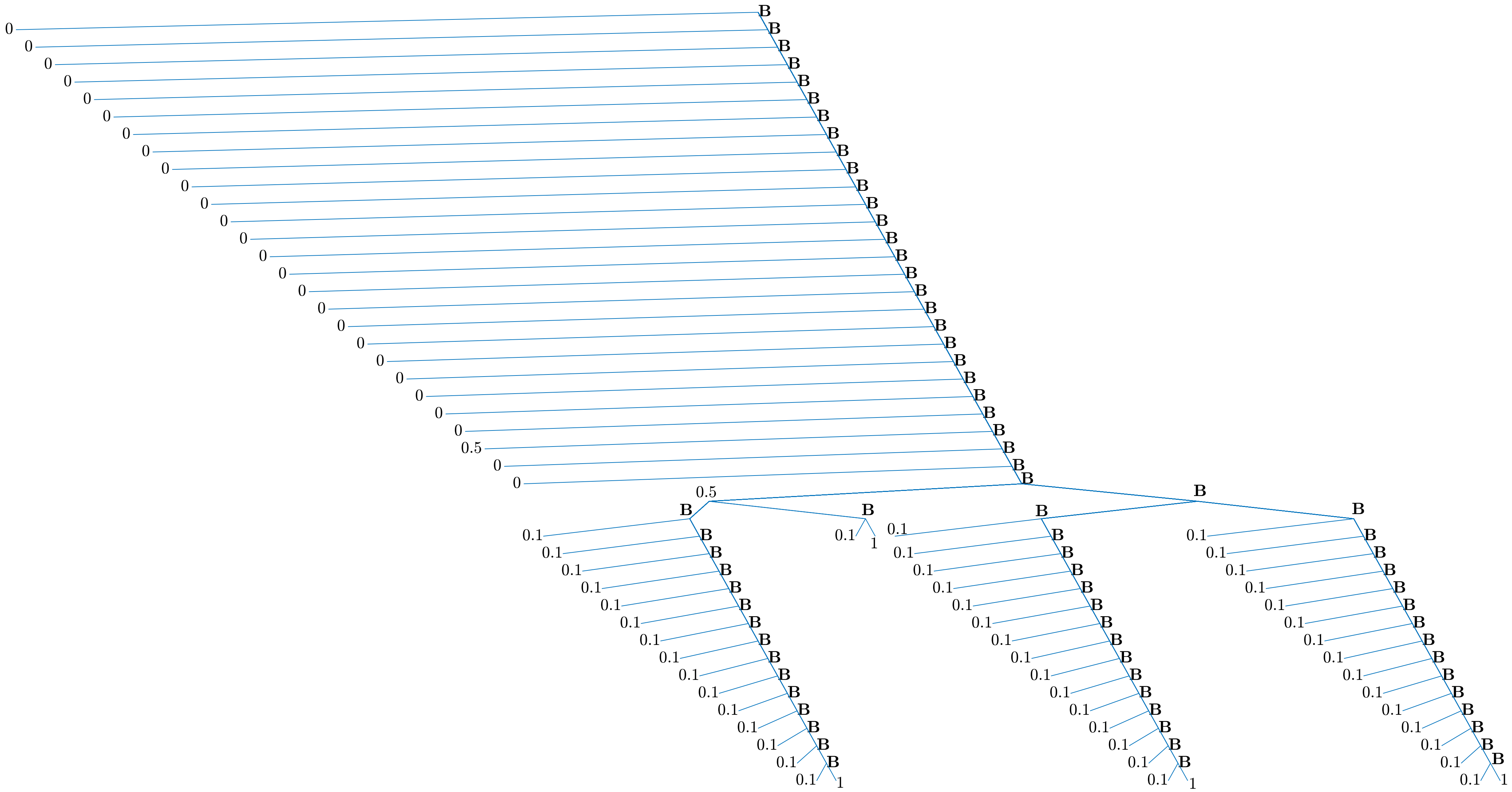


Figure 3: Search tree for `file_rpms_wcnf_L2_V150_C2500_H150_4.wcnf`

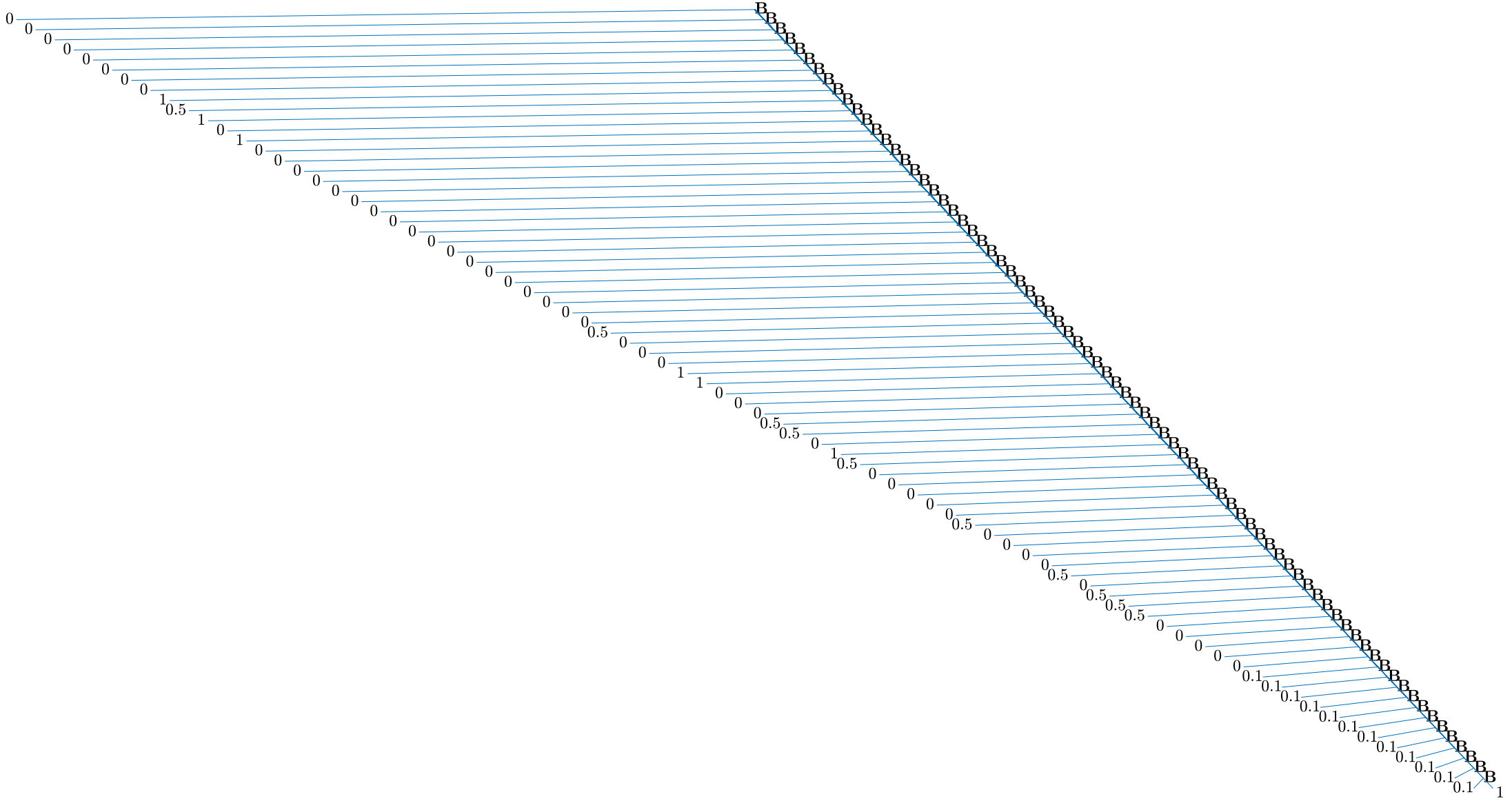


Figure 4: Search tree for `file_rwpms_wcnf_L2_V150_C1000_H150_1.wcnf`

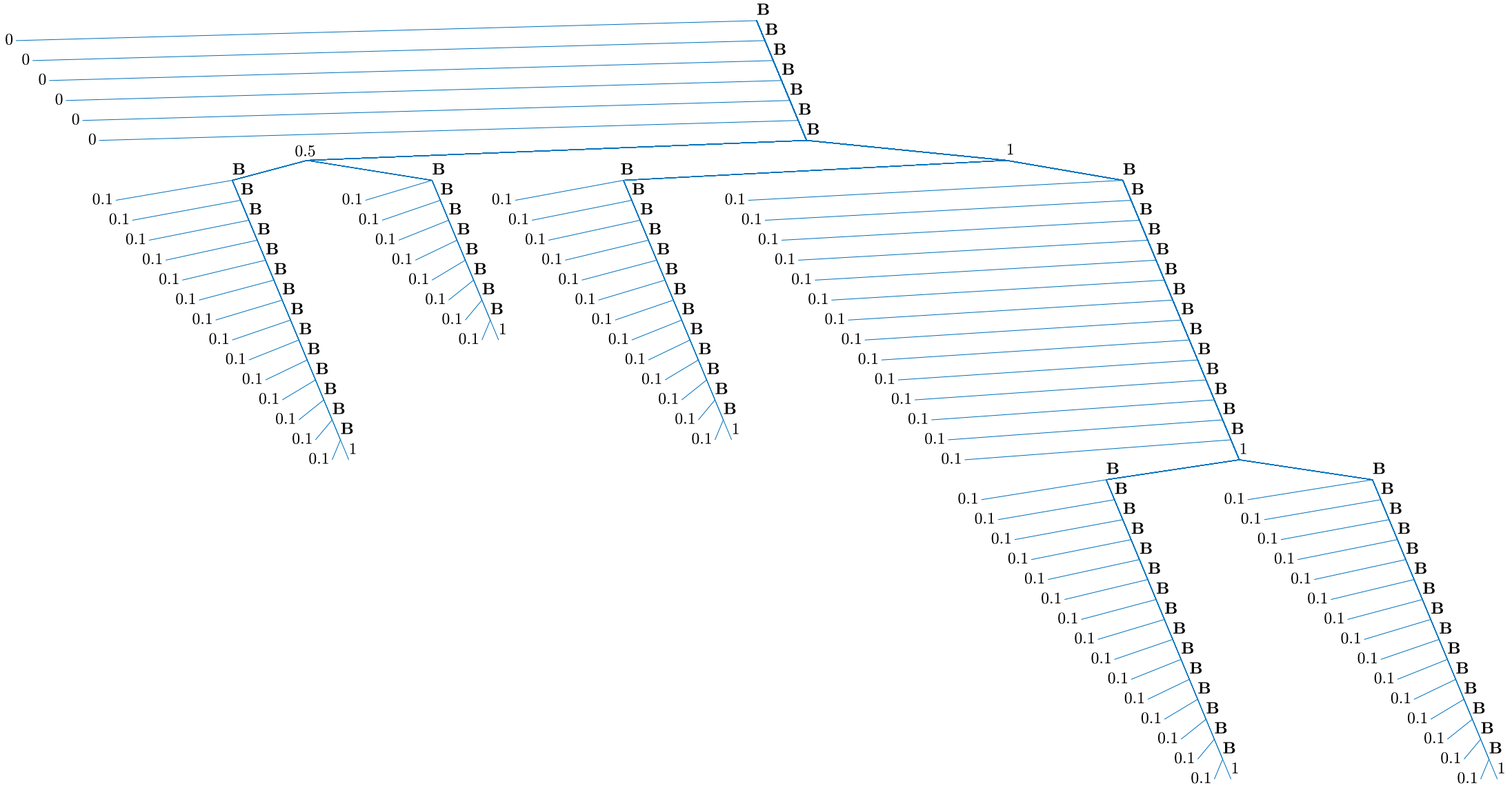


Figure 5: Search tree for `file_rpms_wcnf_L2_V150_C4000_H150_3.wcnf`