The Proof-Search Problem between Bounded-Width Resolution and Bounded-Degree Semi-Algebraic Proofs*

Albert Atserias

Universitat Politècnica de Catalunya
Barcelona, Spain

Abstract. In recent years there has been some progress in our understanding of the proof-search problem for very low-depth proof systems, e.g. proof systems that manipulate formulas of very low complexity such as clauses (i.e. resolution), DNF-formulas (i.e. R(k) systems), or polynomial inequalities (i.e. semi-algebraic proof systems). In this talk I will overview this progress. I will start with bounded-width resolution, whose specialized proof-search algorithm is as easy as uninteresting, but whose proof-search problem is unintentionally solved by certain versions of conflict-driven clause-learning algorithms with restarts. I will continue with R(k) systems, whose proof-search problem turned out to hide the complexity of certain two-player games of interest in the area of systems synthesis and verification. And I will close with bounded-degree semi-algebraic proof systems, whose proof-search problem turned out to hide the complexity of systems of linear equations over finite fields, among other problems.

1 Introduction

Let P be a propositional proof system, which we think of, abstractly, as a polynomial-time verifiable relation between *tautologies* and *proofs*, or dually, between *contradictions* and *refutations* [21]. The proof-search problem for P asks, for a given tautology as input, to find one of its P-proofs. However, since we cannot expect all tautologies to have polynomial-size P-proofs (as this would imply NP = co-NP), we will feel satisfied if we are able to find P-proofs that are not too far from optimal. More formally, a proof system P is called *automatizable in time* t if there exists an algorithm that, when it is given a tautology as input, finds one of its P-proofs in time $t(s)$, where s is the size of its smallest P-proof. Note that we do not insist that the found proof is the shortest possible [17].

The question whether there is an interesting proof system that is automatizable in polynomial time is open. The admittedly vague term *interesting* should mean that the proof system is powerful enough to admit *some* short proofs. For (a non-)example, the proof system whose proof for a given tautology is its full truth-table is *not* interesting for it does not have short proofs at all. This

* Research partially supported by project TIN2010-20967-C04-05 (TASSAT).

makes it trivially automatizable in polynomial time, but for a silly reason. For contrast, interesting proof systems in this sense do include propositional resolution, for example, whose reasoning power is able to produce short proofs of non-trivial tautologies arising in multiple application contexts. For example, resolution admits polynomial-size proofs of the least-number principle (every finite linear order has a least element) [40], which underlies many inductive proofs.

The purpose of this paper is to discuss the status of the proof-search problem for inference-based proof systems that work with formulas of very low complexity. These include resolution or DNF-resolution, which work with clauses and DNF-formulas, respectively, and semi-algebraic proofs, which work with polynomial inequalities over the reals. We also take the opportunity to discuss the connection to some of the lift-and-project methods in mathematical programming.

2 Inference-based proof systems

Most classical proof systems are inference-based: starting with a set of given *hypotheses*, some *conclusions* are produced syntactically by means of one or more inference rules, which are then added to the set of hypotheses to proceed. In producing *proofs* for a tautology, an inference-based proof system starts with the empty set of hypotheses and the goal is to produce the tautology. Of course this will mean that the set of inference rules includes some *axioms*, i.e. inference rules that can be fired without any hypotheses. In producing *refutations* for a contradiction, an inference-based proof system starts with the given contradiction and the goal is to produce some blatant inconsistency.

All typical inference-based proof systems manipulate some particular type of formulas, be them clauses, DNF or CNF-formulas, propositional formulas of some higher but fixed depth of alternations between disjunctions and conjunctions, general propositional formulas, polynomial equations over some ring, polynomial inequalities over some ordered ring, disjunctions of those, decision trees branching on variables or more complicated formulas, binary decision diagrams of various sorts, Boolean circuits, etc. The inference rules are typically some more or less obvious, non-interesting, and polynomially checkable ways of producing some logical consequence of the hypotheses. In this sense, what makes an inference-based proof system more or less powerful is the expressive power of the type of formulas it manipulates.

2.1 Systems that manipulate propositional formulas

In resolution, the formulas are *clauses*, disjunctions of variables or negated variables, and the only inference rule is the *resolution rule*:

$$\frac{A \vee x \qquad B \vee \neg x}{A \vee B},$$

where A and B are clauses and x is a variable. We will see this proof system as a special case of a proof system that manipulates arbitrary propositional formulas

and that has the following inference rules:

$$\frac{}{A \vee \overline{A}} \qquad \frac{A}{A \vee B} \qquad \frac{A \vee C \quad B \vee D}{A \vee B \vee (C \wedge D)} \qquad \frac{A \vee C \quad B \vee \overline{C}}{A \vee B},$$

where A, B, C and D denote propositional formulas in *negation normal form* (i.e. all its negations appear in front of variables), and a bar on top of a formula denotes its *dual* (i.e. $\overline{A \vee B} = \overline{A} \wedge \overline{B}$, $\overline{A \wedge B} = \overline{A} \vee \overline{B}$, $\overline{x} = \neg x$, and $\overline{\neg x} = x$). The four rules above are called *axiom*, *weakening*, *introduction of conjunction*, and *cut*. Besides these rules, the proof system is allowed to produce *structural* manipulations, which means that it is allowed to rewrite a propositional formula into an equivalent one that is obtained by repeated applications of the straightforward rules of commutativity, associativity, and idempotency of disjunctions and conjunctions. We refer to this proof system as F, for *Frege system* [21].

The proof system F is *implicationally complete*, which means that if A is a logical consequence of $A_1, \ldots, A_m$, then there is an F-proof that takes $A_1, \ldots, A_m$ as hypotheses and produces A as conclusion. By the classical results of Cook and Reckhow [21], the reasoning power of F is hence equivalent to any other *Frege proof system*, i.e. any Hilbert-style textbook proof system for propositional logic, and also equivalent to the propositional sequent calculus. By this we mean that every proof in any one of these proof systems can be converted to an F-proof in polynomial time on the size of the proof, and conversely. Here, the size of a proof is the sum of the sizes of the formulas that make it (this includes all the hypotheses, and of course the conclusion). When such efficient conversions from P-proofs into P'-proofs are possible we say that P' *polynomially simulates* P.

As said, resolution can be seen as the special case of this proof system in which the only allowed formulas are clauses and the only allowed rule is cut. When the only allowed formulas are k-DNF-formulas, i.e. disjunctions of conjunctions of up to k literals, the corresponding restriction has been named R(k) or k-DNF-resolution [29]. It is not hard to see that R(1) is equivalent to resolution. When the only allowed formulas are arbitrary DNF-formulas, the proof system is called DNF-resolution.

2.2 Tree-like, dag-like, and bounded-width proofs

An essential feature of inference-based proofs as defined up to now is that, as soon as a conclusion is derived, it can be used multiple times as a hypothesis at no additional cost. On the other hand, it is obvious that every multiple use of a derived hypothesis could be replaced by multiple proofs of that hypothesis, from which it looks like the feature is not that essential after all. However, the point is that in doing this conversion, the proof-size could get exponentially bigger because at every re-derivation we could be doubling the size of the proof up to that point. In the following, we say that a proof in an inference-based proof system is in *tree form*, or *tree-like*, if every derived formula is used at most once as the hypothesis of an inference. Sometimes we use the term *dag form*, or *dag-like*, to emphasize the fact that a certain proof is not in tree form.

The intuition that the dag form of proofs is an essential feature that could lead to exponential savings is indeed correct, but only for proof systems that work with formulas of very low complexity. As will appear clear soon in this section, dag-like proofs can usually be converted efficiently into tree-like proofs whose lines are disjunctions of formulas of the starting dag-like proof or their negations. In particular, this means that in any Frege system such as F, the tree-like and dag-like versions polynomially simulate each other [30]. On the other hand, for proof systems such as resolution or R(k) with constant k, it is known that dag-like proofs could be exponentially shorter [15], [11], [25].

Since for resolution and R(k) tree-like proofs are much less powerful than their dag-like versions, an obvious question arises: why do we even consider the tree-likeness restriction at all? The answer is to be found on the fact that tree-like proofs appear naturally as the result of backtracking procedures. For example, the straightforward backtracking procedure to verify that a given set of clauses is contradictory by branching on the truth values of unset variables, and by pruning each branch as soon as some clause is falsified by the assignment of that branch, corresponds to a tree-like refutation in resolution: turn the recursion tree upside-down, label each leaf by one of the falsified clauses, and label the internal nodes of the tree by a resolution inference on the branched variable [6].

At this point we can ask for the proof system that corresponds to backtracking procedures that branch on the truth value of more complicated formulas and that stop as soon as the assigned truth values incurs into a blatant contradiction with the semantics of the connectives (for example by assigning $A \wedge B$ to true but A to false), or with the given clauses. The correspondence with natural tree-like proofs persists. For example, if the branching formulas are conjunctions of up to k literals, what we get is equivalent to tree-like R(k) [25].

Interestingly, tree-like R(k)-proofs appear naturally in a different context. Suppose $C_1, \ldots, C_m, C_{m+1}, \ldots, C_t$ is a resolution proof of C_t from $C_1, \ldots, C_m$ in which every clause has at most k literals; in that case we say that the resolution proof has *width* k. Since the resolution rule is sound, the last inference-step in this proof is indeed a tautology of the form $C_{\ell(t)} \wedge C_{r(t)} \rightarrow C_t$, or equivalently $\overline{C_{\ell(t)}} \vee \overline{C_{r(t)}} \vee C_t$, where $0 < \ell(t) < r(t) < t$. This tautology depends on no more than $3k$ variables and is a k-DNF, and hence has a tree-like R(k)-proof of size $2^{O(k)}$, and indeed size $O(k)$ because it has very special form. Of course, this is also the case for any inference in the proof. Now, starting at the tautology that corresponds to the inference that derives C_t, and cutting it with the tautologies that correspond to the inferences that derive $C_{\ell(t)}$ and $C_{r(t)}$, we get a k-DNF of the form $\overline{C_{\ell(\ell(t))}} \vee \overline{C_{r(\ell(t))}} \vee \overline{C_{\ell(r(t))}} \vee \overline{C_{r(r(t))}} \vee C_t$. Repeating for every inference in the proof we get $\overline{C_1} \vee \cdots \vee \overline{C_m} \vee C_t$, from which C_t follows by m cuts with the m initial clauses $C_1, \ldots, C_m$. Observe that the result is a tree-like R(k)-proof whose size is a factor $O(k)$ bigger than the original proof (and note also that this argument works equally well to polynomially simulate dag-like F-proofs by tree-like F-proofs [30]).

As we will see later on, the width of a resolution proof as defined in the beginning of the previous paragraph is a very important parameter for the un-

derstanding of resolution. For this reason, let us write R_k for the restriction of resolution in which all clauses have at most k literals. Note that R_k is obviously a restriction of R(1), and from the above, it can also be thought as a restriction of tree-like R(k). However, let us also note that, for $k < n$, the restriction R_k need not be complete on sets of clauses with n variables. Certainly R_k with $k < n$ cannot derive any clause with $k + 1$ literals, or cannot even start if the initial set of clauses contains one with $k + 1$ literals, but even explicit n-variable contradictory sets of 3-clauses are known for which all resolution refutations must use a clause with $\Omega(n)$ literals [12].

2.3 Systems that manipulate polynomial inequalities

If we represent *true* by 1 and *false* by 0, propositional clauses are obviously represented by linear inequalities over the reals. For example, the clause $x \vee \overline{y} \vee z$ is represented by the linear inequality $x + (1 - y) + z \geq 1$, which may be rewritten as $x - y + z \geq 0$. In this sense, resolution may be seen as a proof system that manipulates linear inequalities of special form, over the reals. There are several ways in which this can be generalized to arbitrary linear inequalities. In the cutting planes proof system [18], seen as a proof system for refuting sets of propositional clauses, the hypotheses are represented by linear inequalities of special form as above, the inequalities $x_i \geq 0$ and $1 - x_i \geq 0$ are added to the set of hypotheses, and arbitrary inequalities with integer coefficients may be inferred by means of *positive linear combinations* and *integer rounding*. Although the published work on the cutting planes proof system is very extensive, in this paper we want to focus on a more general family of proof systems that manipulates inequalities over the reals that we call semi-algebraic proof systems.

In the most general semi-algebraic proof system the primary objects are arbitrary polynomial inequalities over the reals. These are inequalities of the form $P \geq 0$, where P is a multi-variate polynomial in the ring of polynomials $\mathbb{R}[x_1, \ldots, x_n]$. The proof system has the following simple rules of inference:

$$\frac{P \geq 0 \quad Q \geq 0}{c \cdot P + d \cdot Q \geq 0} \qquad \frac{P \geq 0 \quad Q \geq 0}{P \cdot Q \geq 0} \qquad \frac{}{P^2 \geq 0}$$

where P and Q are polynomials, and c and d are positive real constants. These rules are called *positive linear combination*, *multiplication rule*, and *positivity of squares*, respectively. Of course a representation issue arises here as the coefficients of the polynomials, as well as the multipliers c and d, could be arbitrary reals. Whenever this issue is important (e.g. when we consider the proof-search problem for such proofs) we will restrict the valid proofs to those that involve rational coefficients that are represented in binary. For the cases of interest, this will not be a severe restriction, as we will see.

Obviously the rules above are sound: if $(x_1, \ldots, x_n) \in \mathbb{R}^n$ satisfies the hypotheses of a rule, then it must also satisfy the conclusion. Moreover, a deep result in real algebraic geometry known as Stengle's Positivstellensatz [41] implies that the rules make up a proof system that is *refutationally complete* for systems

of arbitrary polynomial inequalities. More precisely, if $P_1, \ldots, P_m$ are polynomials in $\mathbb{R}[x_1, \ldots, x_n]$ such that the system $P_1 \geq 0, \ldots, P_m \geq 0$ is unfeasible over $\mathbb{R}^n$, then there is a proof of $-1 \geq 0$ from the hypotheses $P_1 \geq 0, \ldots, P_m \geq 0$ (see [35], [13]). We should also point out that this proof system is *not* implicationally complete for arbitrary polynomial inequalities (see [35], [13] again). As will be evident in the forthcoming, it is often convenient to restrict the degree of all the polynomials appearing in the proof to some bound k. The semi-algebraic proof system restricted to using polynomials of degree at most k will be called $\mathrm{S}^+(k)$. The version without positivity of squares is called $\mathrm{S}(k)$. Let us note that very closely related proof system were called $\mathrm{LS}^k_{+,*}$ and LS^k_* in [27].

Just as clauses can be represented by linear inequalities, there is an obvious way of representing k-DNF-formulas as polynomial inequalities of degree k through *sums of extended monomials*, i.e. inequalities of the form

$$\sum_{t=1}^{m} \prod_{i \in I_t} x_i \prod_{i \in J_t} (1 - x_i) \geq 1$$

where $|I_t \cup J_t| \leq k$ for every $t \in \{1, \ldots, m\}$. Moreover, it is a rather pleasant fact that, under this translation plus some additional axioms stating that $0 \leq x_i \leq 1$ and $x_i^2 = x_i$, the system $\mathrm{R}(k)$ is polynomially simulated by $\mathrm{S}(2k)$ (note $2k$ vs. k). More precisely, if $A_1, \ldots, A_m$ and A are k-DNF-formulas and there is an $\mathrm{R}(k)$-proof of A from $A_1, \ldots, A_m$ of size s, then there is a semi-algebraic proof (of the translation) of A from (the translations of) $A_1, \ldots, A_m$ and the additional axioms $x_i \geq 0$, $1 - x_i \geq 0$, $x_i - x_i^2 \geq 0$ and $x_i^2 - x_i \geq 0$, all whose polynomials have rational coefficients, degree at most $2k$, and the total size of the proof is polynomial in s. The proof of this is not completely trivial, so we give a sketch.

We start by noting that there is a small degree-$2k$ proof of $A + B \geq 1$ from $A + \prod_{i \in I} x_i \prod_{j \in J}(1 - x_j) \geq 1$ and $B + \sum_{i \in I}(1 - x_i) + \sum_{j \in J} x_j \geq 1$ for every two sums of degree-k extended monomials A and B, and $|I \cup J| \leq k$. First observe that if M is a degree-k extended monomial then, in the presence of the four axioms stating $0 \leq x_i \leq 1$ and $x_i^2 = x_i$, there are small degree-$2k$ proofs of $0 \leq M \leq 1$ and $M^2 = M$. In what follows, write $M(I, J)$ for the extended monomial $\prod_{i \in I} x_i \prod_{j \in J}(1 - x_j)$. Now take the second hypothesis $B + \sum_{i \in I}(1 - x_i) + \sum_{j \in J} x_j \geq 1$ and, iteratively for each $i \in I$, multiply by $x_i \geq 0$ and then eliminate $(1 - x_i)x_i$ using $x_i^2 = x_i$. Continuing, iteratively for each $j \in J$, multiply the result by $1 - x_j \geq 0$ and eliminate $x_j(1 - x_j) \prod_{i \in I} x_i$ using $x_j^2 = x_j$ and hence $x_j^2 \prod_{i \in I} x_i = x_j \prod_{i \in I} x_i$. The result is $B \cdot M(I, J) \geq M(I, J)$. Add this to the first hypothesis to get $A + B \cdot M(I, J) \geq 1$. Now, using the fact that B is a sum of extended monomials, derive $B \geq 0$. Derive also $1 - M(I, J) \geq 0$, and multiply together to get $B - B \cdot M(I, J) \geq 0$. Adding this to the above we get $A + B \geq 1$.

The derivation above allows the simulation of cuts except that we also need contraction of repeated terms. In other words, we need small degree-$2k$ proof of $A + Q \geq 1$ from $A + 2Q \geq 1$ for every sum of degree-k extended monomials A and every degree-k extended monomial Q. Proceed as follows: Multiply $A + 2Q \geq 1$ by $1 - Q \geq 0$ to get $A + 3Q - 1 - AQ - 2Q^2 \geq 0$. Then use the fact that A

is a sum of degree-k extended monomials to get $A \geq 0$ and hence $AQ \geq 0$ by multiplication, and add it to the previous inequality to get $A+3Q-1-2Q^2 \geq 0$. Using $Q^2 = Q$ we get $A + Q \geq 1$.

We leave the simulation of axioms, weakenings, and introductions of conjunctions as exercises.

2.4 Connection with lift-and-project methods

In linear programming we are given a collection of linear inequalities $L_1 \geq 0, \ldots, L_m \geq 0$ that define a polyhedron over $\mathbb{R}^n$ and we are asked to optimize a linear function L over the polyhedron. Proving that the optimum is at least some bound c is of course an instance of the general problem of the previous section: prove that $L \geq 0$ follows from given assumptions $L_1 \geq 0, \ldots, L_m \geq 0$, over $\mathbb{R}^n$. However, $L_1, \ldots, L_m$ and L are all linear, and in this case the fundamental duality theorem for linear programming implies that, whenever the implication holds, there is a *linear programming proof*, i.e. one that derives the conclusion as a positive linear combination of the hypotheses and the trivial inequality $1 \geq 0$. Moreover, any polynomial-time algorithm for linear programming can be used to find the proof (by solving the dual).

All this is very good but not directly suited to an arbitrary combinatorial problem in which the implications that matter are over a discrete domain, such as $\{0,1\}^n$, instead of $\mathbb{R}^n$ or $[0,1]^n$. Of course, the domain $\{0,1\}^n$ can be enforced by adding the quadratic constraints $x_i^2 - x_i \geq 0$ and $x_i - x_i^2 \geq 0$, but now, if we want to make use of these constraints, we are forced to go beyond positive linear combinations and use some multiplications or squares. The lift-and-project method of Lovász and Schrijver [33] allows these rules but only in the following limited forms:

$$\frac{P \geq 0 \quad Q \geq 0}{c \cdot P + d \cdot Q \geq 0} \qquad \frac{L \geq 0}{L \cdot x_i \geq 0} \qquad \frac{L \geq 0}{L \cdot (1 - x_i) \geq 0} \qquad \frac{}{L^2 \geq 0}$$

where P and Q are polynomials, L is linear, and c and d are positive real constants. The second and third rules are called *lifting rules*. Besides these rules, the axioms $x_i \geq 0$, $1 - x_i \geq 0$, $x_i - x_i^2 \geq 0$ and $x_i^2 - x_i \geq 0$ are always present (note also that by adding the first two axioms we get $1 \geq 0$).

The proof system introduced by Lovász and Schrijver is called LS^+ in the literature. The version in which positivity of squares is not allowed is called LS. Note that LS and LS^+ are restrictions of $\mathrm{S}(2)$ and $\mathrm{S}^+(2)$, respectively. It is also known that LS polynomially simulates resolution [38]. The restrictions of LS and LS^+ to *lifting rank* less than k are denoted by LS_k and LS_k^+, respectively. Here, the *lifting rank* of a proof is the maximum number of applications of the lifting rules in a path from the hypotheses to the conclusion. Let us note that for $k < n$, the restrictions LS_k and LS_k^+ are not complete over $\{0,1\}^n$; in other words, there exist linear inequalities $L_1, \ldots, L_m$ and L with n variables such that $L \geq 0$ follows from $L_1 \geq 0, \ldots, L_m \geq 0$ over $\{0,1\}^n$, but LS_k and LS_k^+ are not able to prove $L \geq 0$ from $L_1 \geq 0, \ldots, L_m \geq 0$. On the other hand, Lovász

and Schrijver argued that LS_n, and hence LS_n^+, is complete for deriving linear inequalities over $\{0,1\}^n$.

The name "lift-and-project" comes from the idea that the linear inequalities that define the initial polyhedron are lifted to linear inequalities over $\mathbb{R}^{n^2}$ (by thinking of each product $x_i x_j$ as a new variable), and projected back to $\mathbb{R}^n$ through linear combinations (by cancelling all products $x_i x_j$) before a new lifting is allowed.

A different lift-and-project method was suggested by Sherali and Adams [39]. Chronologically, this came before Lovász and Schrijver, but for the purposes of exposition it makes more sense to reverse the order. In the method of Sherali and Adams, the liftings are more powerful, but the way they are combined together is more restricted. Precisely, instead of lifting linear inequalities by multiplication by one literal, we allow lifting of arbitrary polynomials:

$$\frac{P \geq 0}{P \cdot x_i \geq 0} \qquad \frac{P \geq 0}{P \cdot (1 - x_i) \geq 0}$$

where P is an arbitrary polynomial. However, the proofs must have a very special form: they start at the given inequalities $L_1 \geq 0, \ldots, L_m \geq 0$, $x_i \geq 0$, $1 - x_i \geq 0$, $x_i - x_i^2 \geq 0$ and $x_i^2 - x_i \geq 0$, perform a few liftings, and combine them by positive linear combinations (with no further liftings). Thus, all liftings come before all positive linear combinations, and positivity of squares is not allowed. This rather special form will look more natural if we think of the Sherali-Adams method as making a single lift-and-project round, instead of making multiple rounds as in LS, but using dimension n^k for some $k \geq 2$ in the middle stage, instead of dimension n^2 as in LS. The restriction of the Sherali-Adams proof system to polynomials that do not exceed degree k is called SA_k. We call SA_k^+ the natural extension in which, besides the initial inequalities, arbitrary squares are also allowed, but again all restricted to degree at most k.

It is not too hard to see that every proof in LS_k or LS_k^+ can be converted, in polynomial time, into a proof in SA_k or SA_k^+ by *moving the liftings up* towards the hypotheses. Note also that SA_k is a restriction of $\mathrm{S}(k)$ but not a restriction of $\mathrm{S}(k-1)$. Compare this with the fact that, since LS_k is a restriction of LS which in turn is a restriction of $\mathrm{S}(2)$, each LS_k is a restriction of $\mathrm{S}(2)$. As for LS_k and LS_k^+, the restrictions SA_k and SA_k^+ are not complete over $\{0,1\}^n$ when $k < n$, but Sherali and Adams proved that SA_n, and hence SA_n^+, is complete for deriving linear inequalities over $\{0,1\}^n$. Of course nothing prevents us from considering a proof system that allows *multiple rounds* of SA_k as a generalization of LS. This would keep it a subsystem of $\mathrm{S}(k)$ and, indeed, if the number of rounds is unbounded, it would make it equivalent for systems of inequalities that include $x_i \geq 0$ and $1 - x_i \geq 0$. The multiple-round version of SA_k was called LS^k in [27].

One last interesting thing to notice is that SA_k polynomially simulates R_k for sets of clauses. This follows from three facts: 1) that every k-clause of the form $\bigvee_{i \in I} \overline{x_i} \vee \bigvee_{i \in J} x_i$ may be represented by a degree-k inequality of the form $0 \geq M(I,J)$, where $M(I,J)$ is the shorthand notation for extended monomials used earlier, 2) that this representation may be obtained from the given form

$\sum_{i\in I}(1-x_i)+\sum_{i\in J}x_i \geq 1$ of a clause by at most k liftings (ignoring terms that are 0 modulo $x_i^2 = x_i$), and 3) that, in this representation, any width-k resolution step may be simulated by addition with a valid inequality of the form $M(I\cup\{i\},J)+M(I',J'\cup\{i\}) \geq M(I\cup I',J\cup J')$, which has an SA_k-proof itself.

Let us note that Sherali and Adams did not phrase their lift-and-project method in terms of inference rules. Also, they did not consider anything like SA_k^+, which is very closely related to the method of Lasserre. See [32] for a comparison of the three methods.

3 The proof-search problem

After this long introduction, we move now to the proof-search problem for the proof systems introduced in Section 2. We start by stating some positive results and their consequences, then we discuss negative (i.e. conditional hardness) results, and we close with some observations concerning the cases in-between.

3.1 Width-related algorithms

The *width* of a clause is defined as the number of literals it has. In the following, a *k-clause* is one of width at most k. A generous bound on the number of k-clauses on a set of n variables is $(2n+1)^k/k! \leq 2(n+1)^k$. In particular this means that if a contradictory set of clauses has a resolution refutation of width k, then it also has one of size $O(k(n+1)^k)$, where n is the number of variables. It also means that if such a refutation exists, then one can be found in time $n^{O(k)}$ by repeatedly resolving upon known clauses provided the result is an as yet unknown k-clause. This solves the proof-search problem for R_k in time $n^{O(k)}$, where n is the number of variables of the given set of clauses.

As we just noticed, small width refutations entail short refutations. One of the fundamental facts about resolution is that a partial converse is also true: building on the work of Clegg, Edmonds, and Impagliazzo [19] and Beame and Pitassi [8], Ben-Sasson and Wigderson [12] proved if a contradictory set of clauses has a resolution refutation of size s, then it also has a resolution refutation of width $O(\sqrt{n\log s}+w)$, where n is again the number of variables, and w is the width of the widest clause in the given set of clauses. To appreciate the depth of this result let us look at the case of polynomial s and constant w. In that case the width becomes $O(\sqrt{n\log n})$ which is very significantly smaller than the maximum possible width n. It is also known that this trade-off is worst-case optimal (up to logarithmic factors): there exist n-variable sets of 3-clauses that have polynomial-size resolution refutations but that do not have resolution refutations of width $o(\sqrt{n})$ (see [16]).

Among other applications, the fundamental size-width tradeoff result for resolution can be used to argue that, for contradictory sets of w-clauses with constant w, resolution is automatizable in non-trivial time. Consider the algorithm that solves the proof-search problem for R_k in time $n^{O(k)}$ and run it on increasing values of k until the empty clause is found. By the size-width tradeoff, k

will not exceed $O(\sqrt{n \log s})$ where s is the size of the shortest resolution refutation (recall that we are assuming that w is a constant, but we could even afford $w = O(\sqrt{n \log n})$ for this to be true). Hence the algorithm runs in time $n^{O(\sqrt{n \log s})}$. Note how this is a non-trivial time-bound: if s is polynomial, the running time is of subexponential type $2^{O(n^{0.51})}$.

The width-based algorithm from the preceding paragraph is not terribly satisfying in that it completely ignores any structure that the input set of clauses could have, and blindly derives all possible clauses (in increasing order of width). In contrast, practically-used resolution-based algorithms exploit very fine-tuned heuristics that *learn* strategically chosen clauses with the hope of deriving the empty clause earlier or pruning the search-space to a point where exhaustive search for a satisfying assignment becomes successful [34]. Of course one could always run the width-based algorithm in parallel to a fine-tuned heuristic-based algorithm in order to guarantee the worst-case bound from the first with the practical features of the second. But, somewhat surprisingly, it turned out that the architecture of most practically-used algorithms does not require this. The relatively recent result from [3] shows that if a standard *conflict-driven clause-learning algorithm* (CDCL algorithm) is given the opportunity to restart and branch on randomly chosen literals often enough, then the resulting algorithm is guaranteed to have high probability of finding a refutation after no more than $n^{O(k)}$ iterations, if a resolution refutation of width k exists. Interestingly, the validity of this result is quite robust to the actual tuning of the underlying CDCL algorithm. We refer the reader to the reference [3] for details.

3.2 Degree-related algorithms

The original motivation for the lift-and-project methods from Section 2.4 was to devise a method by which an initial polytope P over $[0,1]^n$ could be *tightened* into better and better approximations $P \supseteq P_1 \supseteq P_2 \supseteq \cdots \supseteq P_n = P^*$, where P^* denotes the convex hull of the 0-1 points in P. Both SA and LS achieve this by letting P_k be the polytope defined by the inequalities that have an SA_k or LS_k-proof from the inequalities that define the initial polytope. Moreover, and this is the main point of the methods, in both cases there is an algorithm running in time $n^{O(k)}$ to optimize any given linear objective function over the polytope P_k (see [39], [33]).

For SA_k even more is true. Not only it is possible to optimize linear functions over P_k, but even SA_k-proofs of optimality can be found. More precisely, there exists an algorithm that, given linear functions $L_1, \ldots, L_m$ and L, finds an SA_k-proof of $L \geq 0$ from $L_1 \geq 0, \ldots, L_m \geq 0$, if there is one, and does so in time $n^{O(k)}$, where n is the number of variables. One way to see this is by first observing that, during the phase of liftings in an SA_k-proof, all we are doing is multiplying the given inequalities and the axioms $x_i^2 - x_i \geq 0$ by an extended monomial of the form $\prod_{i \in I} x_i \prod_{j \in J}(1 - x_j)$ with $|I \cup J| \leq k$, of which there are no more than $(2n)^k$. The second observation is that what is left to do in the phase of positive linear-combinations is a linear programming problem over $\mathbb{R}^{(n+1)^{k+1}}$

(by interpreting each monomial of degree at most $k+1$ as a new variable). Thus, any algorithm solving linear programming in polynomial time will give an algorithm to solve the proof-search problem for SA_k in time $n^{O(k)}$. Observe that, by a simple binary search argument, this is a stronger claim than the ability to optimize over the polytope P_k.

For the lift-and-project method LS_k of Lovász and Schrijver only the weaker claim about optimization is known. The difficulty in providing explicit LS_k-proofs of optimality is that the optimization algorithm works by providing a polynomial-time separation oracle for the polytope P_{i+1} given a separation oracle for P_i, and using this recursion to apply the ellipsoid method on P_k. An intriguing observation is that if we are happy with an SA_k-proof of optimality, then we can still get it time $n^{O(k)}$. The reason for this is that, as mentioned in Section 2.4, there is a polynomial translation of LS_k-proofs into SA_k-proofs. Thus, the algorithm from the previous paragraph applies. This also shows that the optimization problem for LS_k can also be solved without resorting to the ellipsoid method.

For SA_k^+ and LS_k^+ similar statements are true by using polynomial-time algorithms for semi-definite programming in one case, and the ellipsoid method in the other. In both cases, the key observation is that the sums of squares of linear forms are in one-to-one correspondance with the positive semi-definite quadratic forms. For these reasons, SA_k^+ and LS_k^+ are called the *semi-definite versions* of SA_k and LS_k. The catchy acronym SoS (for sum-of-squares) is also used for certain versions of SA_k^+ (see [35], [5]).

3.3 Reductions from tree-form to bounded width or degree

The version of the Ben-Sasson-Wigderson size-width tradeoff for tree-like resolution is this: if a set of clauses has a tree-like resolution refutation of size s, then it also has a resolution refutation of width $O(\log s + w)$, where w is the width of the largest clause in the given set of clauses. In particular, this means that for constant w, by running the proof-search algorithm for R_k with increasing values of k until we find the empty clause, we succeed in time $n^{O(\log s)}$, where s is the size of the shortest tree-like refutation and n is the number of variables. Note however that the proof is not necessarily tree-like. In other words, the algorithm runs within a non-trivial time-bound that depends on the size of the shortest tree-like refutation, but the obtained proof is in a different proof system.

For tree-like LS and LS^+ what happens is closer to what happens for dag-like resolution. The analogue size-rank tradeoff for tree-like LS and LS^+ was shown by Pitassi and Segerlind [36]: if a system of linear inequalities with n variables has a tree-like LS-refutation of size s, then it also has an LS_k-refutation with $k = O(\sqrt{n \log s})$, and the same for LS^+. Again this gives an algorithm that, given a system of linear inequalities that is contradictory over $\{0,1\}^n$, finds an LS-refutation in time $n^{O(\sqrt{n \log s})}$, where s is the size of the shortest tree-like LS-refutation, and n is the number of variables. However, the obtained proof is not necessarily in tree form.

When this happens, namely that there is an algorithm that given a tautology finds one of its P'-proofs in time $t(s)$ where s is the size of the smallest P-proof, we say that P is *weakly automatizable* (in terms of P') in time t. For later use, let us point out that it is not hard to see that P is weak automatizable in polynomial time if and only if there is a proof system P' that is automatizable in polynomial time and that polynomially simulates P [37].

Using the fact that $s \geq n$ because the hypotheses are counted in the size of any proof, what the first paragraph of this section says is that tree-like resolution is weakly automatizable in quasi-polynomial time of the type $s^{O(\log s)}$. For the sake of completeness, let us also mention that for tree-like resolution, a direct (non-weak) proof-search algorithm that runs in quasi-polynomial time $s^{O(\log s)}$ is known [8]. The latter has the added advantage that it works for arbitrary sets of clauses and not only those of limited width.

3.4 Hardness results

The weak automatizability of a proof or refutation system is closely related to the concept of *feasible interpolation* [31], [37]. In short, the interpolation problem for a refutation system P is the following: given a P-refutation of a conjunction $A_0 \wedge A_1$, where A_0 and A_1 are formulas on disjoint sets of variables, output $b \in \{0, 1\}$ such that the formula A_b is contradictory by itself. Under a very mild closure condition on the set of P-refutations, the connection is that if P is weakly automatizable in polynomial time, then the interpolation problem for P can also be solved in polynomial time. The mild closure condition, called *natural* in [7], is that if a contradictory formula A has a P-refutation of size at most s, then the result of assigning any truth value to any one of the variables of A also has a P-refutation of size at most s. This is true of virtually any refutation system one can think of (but see [7] where it is pointed out that this is not so clear for proofs produced by CDCL algorithms).

To see the connection pointed out above argue as follows: Let P' be the refutation system that is automatizable in polynomial time and that polynomially simulates P. Given a P-refutation of $A_0 \wedge A_1$ as input, first we run the proof-search algorithm for P' on input A_0 until either it finds a P'-refutation or it runs for more than $t(p(s))$ steps, where s is the size of the given P-refutation of $A_0 \wedge A_1$, and t and p are, respectively, the polynomials that bound the running time of the proof-search algorithm for P', and the size of the P'-refutations as a function of the size of the P-refutations. In the first case we output 0. In the second case we know that A_1 cannot be satisfiable and it is safe to output 1 (otherwise, by the mild closure condition, plugging one of its satisfying assignment into the P-refutation of $A_0 \wedge A_1$ would give a size-s P-refutation of A_0). See [37], [2] for more on this.

Several interesting proof systems have feasible interpolation, which means that their interpolation problem can be solved in polynomial time. These include resolution, cutting planes, and LS [31], [38]. On the other hand, if we want to show that a proof system P is not automatizable in polynomial time, it suffices to argue that it does not have feasible interpolation. Typically this is done by

reducing a (conjecturally) hard problem to the interpolation problem for P. More precisely, starting at a problem for which distinguishing the YES-instances from the NO-instances requires more than polynomial time, we want to find a polynomial-time translation from instances into P-refutations of certain formulas of the form $A_0 \wedge A_1$, in such a way that YES-instances give a satisfiable A_0 and NO-instances give a satisfiable A_1.

This strategy for arguing the failure of feasible interpolation can be made to work for several proof systems. For example, in the same paper where the concept of automatization was defined, Bonet, Pitassi and Raz followed this strategy to prove that no Frege system P has feasible interpolation unless factoring Blum integers can be solved in polynomial time. Intuitively, the formula $A_0 \wedge A_1$ states that a *hard bit* of the given Blum integer is both 0 and 1 at the same time, and a short P-proof is given for the impossibility of this fact. This established that no Frege system is automatizable or weakly automatizable in polynomial time under a reasonable cryptographic conjecture. Of course, this applies as well to the Frege system F from Section 2.1. A few years later the argument was refined to prove a weaker negative result for Frege-systems working with formulas of fixed (but large) AND/OR alternation depth [14]. It was shown that for every large enough depth d, such systems do not have feasible interpolation unless factoring Blum integers can be solved in subexponential time $2^{n^{1/d^{O(1)}}}$.

For resolution and other *low-depth* proof systems such as R(k) or S(k) for constant k, LS, or DNF-resolution, the situation is less clear. Part of the difficulty is that resolution and LS *do* have feasible interpolation and therefore the type of arguments above cannot be made to work. Thus, if we want to make progress in our understanding of the automatizability of resolution we need to focus on the proof-search problem itself. That is what Alekhnovich and Razborov did, i.e. they reduced a conjecturally hard problem to the problem of distinguishing formulas with small resolution refutations from formulas that do not have much larger resolution refutations [1]. This way they proved that resolution and tree-like resolution are not automatizable in polynomial time unless W[P] is tractable. Without entering the details of W[P], let us mention that the intractability of W[P] is quite likely as otherwise it would mean that the k-clique problem on graphs with n vertices can be solved by a probabilistic algorithm in time $f(k) \cdot p(n)$ for some fixed computable function f independent of n and some fixed polynomial p independent of k (see [22]).

3.5 Games and propositional proofs

The hardness result of Alekhnovich and Razborov says nothing about the possibility that resolution could be automatizable in quasi-polynomial time. Indeed, as mentioned in Section 3.3, tree-like resolution *is* automatizable in quasi-polynomial time and this is not incompatible with the result of Alekhnovich and Razborov. Also, it says nothing about the possibility that resolution or tree-like resolution could be weakly automatizable in polynomial time. In particular, it says nothing about the possibility that any of R(k) or S(k) for $k \geq 2$, LS or cutting planes, or DNF-resolution could be automatizable in polynomial time. All

these are important proof systems for which their automatizability could be an important breakthrough. In view of this, since the proof-search problem is about distinguishing formulas with small proofs from those without, it makes sense to continue the search for large families of formulas that admit small proofs in these systems. With this in mind, a line of recent work has uncovered an interesting connection between some classical problems in game theory and some of these low-depth proof systems. We discuss this in the rest of this section.

In a mean-payoff game (MPG) two players take rounds at extending a path on a finite weighted directed graph. The game ends as soon as the path intersects itself, forming a cycle. The first player wins if the average weight of the cycle is positive. Otherwise the second player wins. Classically, the game is played for infinitely many rounds, but for our purposes this finite version suffices [23], [43]. The problem of mean-payoff games asks, for a given game graph with weights written in binary, whether the first player has a winning strategy. The exact complexity of this problem is unknown, but is known to lie in NP $\cap$ co-NP (see [43]). The connection with low-depth proof systems was found by Atserias and Maneva who showed that every MPG can be converted in polynomial time into a set of clauses that is either satisfiable, in which case the first player has a winning strategy, or has a polynomial-size DNF-refutation, in which case the second player has a winning strategy [4]. In particular, this shows that if DNF-resolution were automatizable or even weakly automatizable in polynomial time, then MPGs would be solvable in polynomial time.

Shortly after this was shown, Huang and Pitassi improved this to the simple stochastic games (SSG) of Condon [20], a class of games to which MPGs reduce in polynomial time. Indeed, their proof showed more since they reduced SSGs to the interpolation problem for DNF-resolution. In the conference version of their paper [28], the reduction was stated to produce depth-3 refutations instead of DNF-resolution-refutations, but it was later pointed out that the refutations are indeed in DNF-resolution. We close this section by describing the latest development in this line of research which takes us to a third type of games.

In a parity game (PG) again two players take rounds at extending a path on a finite directed graph, this time unweighted. The game ends as soon as the path intersects itself, forming a cycle. The first player wins if the least numbered vertex in the cycle is odd. Otherwise the second player wins. As with MPGs, classically the game is played indefinitely, but for us the finite version will be enough. The problem of parity games asks for the winner of a given PG. Parity games have their origins in automata theory where they are used to give combinatorial semantics to the modal μ-calculus, among other things [24]. Again the complexity of the problem of PGs is unknown, but it is known to reduce to MPGs and in particular belongs to NP $\cap$ co-NP (see [43]).

An interesting recent discovery of Beckmann, Pudlák and Thapen [9] is that the problem of parity games (PG) reduces to the interpolation problem for R(k) for a fixed constant $k \geq 2$. In particular, by known results relating interpolation of R(k) with weak automatizability of resolution (see [2]), this means that if resolution were weakly automatizable, then PGs would be solvable in polynomial

time. This last possibility is not fully unlikely, even conjectured by some, but at least it shows that the proof-search problem for resolution must be at least as hard as PGs, a notorious 20 year-old unsolved problem. It also reinforces the claim that resolution, and R(k) with constant k, are "interesting" from the point of view of the proof-search problem in the sense it was meant in the introduction.

3.6 Semi-algebraic proofs and linear equations mod 2

Note that the results mentioned in the last paragraph of the previous section also apply to the semi-algebraic systems LS and S(k) for $k \geq 2$. This is because the results were referring to weak automatizability of resolution, and the systems LS and S(2) polynomially simulate resolution. We want to finish this paper by pointing out what we believe is an important characteristic that distinguishes these semi-algebraic systems from resolution and even DNF-resolution.

Hand-in-hand with the pigeonhole principle, unsolvable systems of linear equations over the 2-element field make one of the classical sources of hardness for resolution-based proof systems, and even bounded-depth Frege systems; the celebrated Tseitin formulas illustrate the point [42], [12], [10]. On the other hand, it was shown by Grigoriev, Hirsch, and Pasechnik [27] that the Tseitin formulas are not hard for a system very related to S(k), for some constant k. Even more, a careful look at their proof shows that any unsolvable system of linear equations mod 2 in which each equation has at most three non-zero coefficients, when appropriately encoded as a set of clauses, has polynomial-size refutations in S(5) by simulating Gaussian elimination. This should be put in contrast with the results of Grigoriev [26] that imply that any SA_k^+-refutation of the Tseitin formulas requires $k = \Omega(n)$, where n is the number of variables.

What these observations say is that the proof-search problem for S(k) for constant $k \geq 5$ hides the complexity of systems of linear equations over the 2-element field. Of course this is not a computationally hard problem since Gaussian elimination solves it in polynomial time. But the point is that if S(k) is to have an efficiently solvable proof-search algorithm, this algorithm will need to be at least as clever as it takes to solve systems of linear equations. In particular, it also says that the distance between SA_k and S(k) is much bigger than it could look from the definitions, and that the methods for solving the proof-search problem for SA_k are probably completely irrelevant to S(k). We would love to be wrong on this and be shown that a clever application of the ellipsoid algorithm, say, is able to lift the proof-search algorithm for SA_k to a linear optimization algorithm for (some interesting version of) S(k).

Acknowledgments We thank the comments of Allen Van Gelder and an anonymous referee on the preliminary draft of this paper.

References

1. M. Alekhnovich and A. A. Razborov. Resolution Is Not Automatizable Unless W[P] Is Tractable. SIAM J. Comput., 38(4), pp. 1347-1363, 2008.

2. A. Atserias and M. L. Bonet. On the Automatizability of Resolution and Related Propositional Proof Systems. Information and Computation, 189(2), pp. 182-201, 2004.
3. A. Atserias, J. K. Fichte and M. Thurley. Clause-Learning Algorithms with Many Restarts and Bounded-Width Resolution. Journal of Artificial Intelligence Research, 40, pp. 353-373, 2011.
4. A. Atserias and E. Maneva. Mean-payoff games and propositional proofs. Information and Computation, 209(4), pp. 664-691, 2011.
5. B. Barak, F. Brandão, A. Harrow, J. Kelner and Y. Zhou. Hypercontractivity, Sum-of-Squares Proofs, and their Applications. In Proc. of 44th ACM Symposium on Theory of Computing (STOC), pp. 307326, 2012.
6. P. Beame, R. Karp, T. Pitassi and M. Saks. The efficiency of resolution and Davis-Putnam procedures. SIAM J. Comput., 31(4), pp. 1048-1075, 2002.
7. P. Beame, H. Kautz and A. Sabharwal. Towards Understanding and Harnessing the Potential of Clause Learning. Journal of Artificial Intelligence Research, 22, pp. 319-351, 2004.
8. P. Beame and T. Pitassi. Simplified and improved Resolution lower bounds. In Proc. of the 27th IEEE Foundations of Computer Science (FOCS), pp. 274-282, 1996.
9. A. Beckmann, P. Pudlák and N. Thapen. Parity games and propositional proofs. In preparation, April 2013.
10. E. Ben-Sasson. Hard examples for the bounded depth Frege proof system. Computational Complexity, 11, pp. 109-136, 2002.
11. E. Ben-Sasson, R. Impagliazzo and A. Wigderson. Near Optimal Separation of Tree-Like and General Resolution. Combinatorica, 24(4), pp. 585-604, 2003.
12. E. Ben-Sasson and A. Wigderson. Short proofs are narrow–resolution made simple. Journal of the ACM, 48(2), pp. 149-169, 2001.
13. J. Bochnak, M. Coste and M.-F. Roy. *Real algebraic geometry*. Springer-Verlag, 1999.
14. M. L. Bonet, C. Domingo, R. Gavaldà, A. Maciel, T. Pitassi. Non-automatizability of bounded-depth Frege proofs. Computational Complexity, 13, pp. 47-68, 2004.
15. M. L. Bonet, J. L. Esteban, N. Galesi and J. Johansen. On the Relative Complexity of Resolution Refinements and Cutting Planes Proof Systems. SIAM J. Comput., 30(5), pp. 1462-1484, 2000.
16. M. L. Bonet and N. Galesi. Optimality of Size-Width Tradeoffs for Resolution. Computational Complexity, 10(4), pp. 261-276, 2001.
17. M. L. Bonet, T. Pitassi and Ran Raz. On Interpolation and Automatization for Frege Systems. SIAM J. Comput., 29(6), pp. 1939-1967, 2000.
18. V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems Discrete Mathematics, 4(4), pp. 305-337, 1973.
19. M. Clegg, J. Edmonds and R. Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In Proc. of the 28th annual ACM Symposium on Theory of Computing (STOC), pp. 174-183, 1996.
20. A. Condon. The Complexity of Stochastic Games. Information and Computation, 96, pp. 203-224, 1992.
21. S. A. Cook and R. A. Reckhow. The Relative Efficiency of Propositional Proof Systems. J. Symbolic Logic, 44(1), pp. 36-50, 1979.
22. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science, Springer-Verlag, 1999.
23. A. Ehrenfeucht and J. Mycielsky. Positional strategies for mean payoff games. International Journal of Game Theory, 8(2), pp. 109-113, 1979.

24. E. A. Emerson and C. S. Jutla. Tree Automata, Mu-Calculus and Determinacy. In Proc. of 32nd IEEE Foundations of Computer Science (FOCS), pp 368-377, 1991.
25. J. L. Esteban, N. Galesi and J. Messner. On the Complexity of Resolution with Bounded Conjunctions. Theoretical Computer Science, 321(2-3), pp. 347-370, 2004.
26. D. Grigoriev. Linear Lower Bound on Degrees of Positivstellensatz Calculus Proofs for the Parity. Theor. Comput. Sci., 259, pp. 613-622, 2001.
27. D. Grigoriev, E. A. Hirsch and D. V. Pasechnik. Complexity of semi-algebraic proofs. Moscow Mathematical Journal 2(4), pp. 647-679, 2002.
28. L. Huang and T. Pitassi. Automatizability and Simple Stochastic Games. In Proc. of 38th International Colloquium on Automata, Languages and Programming (ICALP), Luca Aceto, Monika Henzinger, Jiri Sgall (Eds.), Zurich, Switzerland, Part I. Lecture Notes in Computer Science 6755, Springer, pp. 605-617, 2011.
29. J. Krajíček. On the weak pigeonhole principle. Fundamenta Mathematicae, 170(1-3), pp. 123-140, 2001.
30. J. Krajíček. Lower Bounds to the Size of Constant-Depth Propositional Proofs. J. of Symbolic Logic, 59(1), pp. 73-86, 1994.
31. J. Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. J. of Symbolic Logic, 62(2), pp. 457-486, 1997.
32. M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for 0-1 programming. Mathematics of Operations Research, 28, pp. 470-496, 2001.
33. L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. SIAM J. Optimization, 1, pp. 166-190, 1991.
34. J. Marques-Silva, I. Lynce and S. Malik. Conflict-Driven Clause Learning SAT Solvers. in *Handbook of Satisfiability*, Armin Biere, Marijn Heule, Hans van Maaren and Toby Walsh (Eds.), IOS Press, 2009.
35. P. A. Parrilo. Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization. Ph.D. Thesis, California Institute of Technology, Pasadena, CA, 2000.
36. T. Pitassi and N. Segerlind. Exponential Lower Bounds and Integrality Gaps for Tree-Like Lovász-Schrijver Procedures. SIAM J. Comput., 41(1), pp. 128-159, 2012.
37. P. Pudlák. On reducibility and symmetry of disjoint NP-pairs. Theor. Comput. Science, 295, pp. 323-339, 2003.
38. P. Pudlák. On the complexity of propositional calculus. *Sets and Proofs, Invited papers from Logic Colloquium'97*, Cambridge Univ. Press, pp. 197-218, 1999.
39. H. D. Sherali and W. P. Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zeroXone programming problems. Discrete Applied Mathematics, 52(1), pp. 83-106, 1994.
40. G. Stålmarck. Short resolution proofs for a sequence of tricky formulas. Acta Informatica, 33(3), pp. 277-280, 1996.
41. G. Stengle. A Nullstellensatz and a Positivstellensatz in Semialgebraic Geometry. Mathematische Annalen, 207(2), pp. 87V97, 1974.
42. A. Urquhart. Hard examples for resolution. Journal of the ACM, 34(1), pp. 209-219, 1987.
43. U. Zwick and M. S. Paterson. The complexity of mean payoff games on graphs. Theoretical Computer Science 158, pp. 343-359, 1996.