Stabbing Planes*†

Paul Beame[1], Noah Fleming[2], Russell Impagliazzo[3], Antonina Kolokolova[4], Denis Pankratov[5], Toniann Pitassi[6], and Robert Robere[7]

1 **University of Washington, Seattle, USA**
`beame@cs.washington.edu`

2 **University of Toronto, Toronto, Canada**
`noahfleming@cs.toronto.edu`

3 **University of California, San Diego, USA**
`russell@cs.ucsd.edu`

4 **Memorial University of Newfoundland, St. John's, Canada**
`kol@mun.ca`

5 **University of Toronto, Toronto, Canada**
`denisp@cs.toronto.edu`

6 **University of Toronto and Institute for Advanced Study, Toronto, Canada**
`toni@cs.toronto.edu`

7 **University of Toronto, Toronto, Canada**
`robere@cs.toronto.edu`

Abstract

We introduce and develop a new semi-algebraic proof system, called Stabbing Planes that is in the style of DPLL-based modern SAT solvers. As with DPLL, there is only one rule: the current polytope can be subdivided by branching on an inequality and its "integer negation." That is, we can (nondeterministically choose) a hyperplane $ax \geq b$ with integer coefficients, which partitions the polytope into three pieces: the points in the polytope satisfying $ax \geq b$, the points satisfying $ax \leq b-1$, and the middle slab $b-1 < ax < b$. Since the middle slab contains no integer points it can be safely discarded, and the algorithm proceeds recursively on the other two branches. Each path terminates when the current polytope is empty, which is polynomial-time checkable. Among our results, we show somewhat surprisingly that Stabbing Planes can efficiently simulate Cutting Planes, and moreover, is strictly stronger than Cutting Planes under a reasonable conjecture. We prove linear lower bounds on the *rank* of Stabbing Planes refutations, by adapting a lifting argument in communication complexity.

1998 ACM Subject Classification F.0 General

Keywords and phrases communication complexity, cutting planes, proof complexity

Digital Object Identifier 10.4230/LIPIcs.ITCS.2018.10

1 Introduction

While defined in terms of non-deterministic algorithms for the Tautology problem, proof complexity has also provided indispensable tools for understanding deterministic algorithms for search problems, and in particular, for Satisfiability algorithms. Many algorithms for

* This work was partially supported by NSERC and by NSF awards CCF-1524246, CCF-1412958, CCF-1213151. Part of the work was done while at Simons institute.

† Full version of this paper is available at[1], `https://arxiv.org/abs/1710.03219`

9th Innovations in Theoretical Computer Science Conference (ITCS 2018).
Editor: Anna R. Karlin; Article No. 10; pp. 10:1–10:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

search can be classified according to the types of reasoning they implicitly use for case-analysis and pruning unpromising branches. Particular families of search algorithms can be characterized by *formal proof systems*; the size of proofs in these formal proof system, the time of the non-deterministic algorithm, captures the time taken on the instance by an *ideal implementation* of the search algorithm. This allows us to factor understanding the power of search algorithms of a given type into two questions:

1. How powerful is the proof system? For which kinds of input are there small proofs?
2. How close can actual implementations of the search method come to the ideal non-deterministic algorithm?

As an illustrative example, let us recall the *DPLL* algorithm [10, 9], which is one of the simplest algorithms for SAT and forms the basis of modern conflict-driven clause learning SAT solvers. Let $\mathcal{F} = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be a CNF formula over variables $x_1, x_2, \ldots, x_n$. A DPLL search tree for solving the SAT problem for $\mathcal{F}$ is constructed as follows. Begin by choosing a variable x_i (non-deterministically, or via some heuristic), and then recurse on the formulas $\mathcal{F} \restriction x_i = 0$, $\mathcal{F} \restriction x_i = 1$. If at any point we have found a satisfying assignment, the algorithm outputs *SAT*. Otherwise, if we have falsified every literal in some clause C, then we record the clause and halt the recursion. If every recursive branch ends up being labelled with a clause and a falsifying assignment, then the original formula $\mathcal{F}$ is unsatisfiable and one can take the tree as a proof of this fact; in fact, such a DPLL tree is equivalent to a *tree-like* Resolution refutation of the formula $\mathcal{F}$.

Modern SAT solvers still have a DPLL algorithm at the core (now with a highly tuned *branching heuristic* that chooses the "right" order for variables and assignments to recurse on in the search tree), but extends the basic recipe in two ways: smart handling of *unit clauses* (if $\mathcal{F}$ contains a clause with a single variable x under the current partial assignment, x can be immediately set so that the clause is satisfied), and *clause learning* to speed up search: if a partial assignment ρ falsifies a clause, then the algorithm derives a new clause C_ρ by a resolution proof that "explains" this conflict, and adds the new clause to the formula $\mathcal{F}$.

It is the synergy between these three mechanisms – branching heuristics, unit propagation, and clause learning – that results in the outstanding performance of modern SAT solvers. In other words, while these algorithms are all formalizable in the same simple proof system, the sophistication of modern SAT-solvers comes from the attempt to algorithmically find small proofs when they exist. In many ways, the simplicity of the proof system enables this sophistication in proof-search methods.

In this work, we introduce a natural generalization of the DPLL-style branching algorithm to reasoning over integer-linear inequalities, formalized as a new semi-algebraic proof system that we call the *Stabbing Planes* (SP) system. We will give a more detailed description later, but intuitively, Stabbing Planes has the same branching structure as DPLL, but generalizes branching on single variables to branching on linear inequalities over the variables. We feel the closeness to DPLL makes Stabbing Planes a better starting point for understanding search algorithms based on linear inequalities, as in integer linear programming (ILP) based solvers, than established proof systems such as Cutting Planes.

We compare the power of Stabbing Planes proofs to these other proof systems. Recall that Cutting Planes (CP) is a proof system for reasoning over linear inequalities using linear combination and division with rounding rules, and Krajícek's system R(CP) combines resolution and CP rules. We show that tree-like R(CP) is polynomially equivalent to Stabbing Planes (Theorem 9). However, the new formulation as Stabbing Planes proofs both gives greater motivation to studying R(CP) and greatly clarifies the power of this proof system. Our main results about this system are:

1. Stabbing Planes has quasi-polynomial size and poly-log rank proofs of any tautology provable using linear algebra over a constant modulus. In particular, this is true for the Tseitin graph tautologies, that are very frequently used to prove lower bounds for other proof systems. (Theorem 6)
2. Stabbing Planes can simulate tree-like Cutting Planes proofs with only constant factor increases in size and rank (Theorem 11), and general Cutting Planes proofs with a polynomial increase in size (Theorem 12)
3. Lower bounds on real communication protocols imply rank lower bounds for Stabbing Planes proofs (Lemma 17)
4. Stabbing Planes proofs cannot be balanced (Theorem 21).

Together, these show that Stabbing Planes is at least as strong as established proof systems using inequalities, and possibly much stronger. So the proof system combines strength as a proof system with a simple branching structure that raises the possibility of elegant algorithms based on this proof system, in particular for pseudoBoolean solvers.

We now give a more precise description of the proof system. Let us formalize the system in stages. First, observe that the setting is quite different: we are given a system $A_1 x \geq b_1, A_2 \cdot x \geq b_2, \ldots, A_m \cdot x \geq b_m$ of integer-linear inequalities over real-valued variables $x_1, x_2, \ldots, x_n$ (for simplicity we will always assume that the inequalities $0 \leq x_i \leq 1$ are present for each variable x_i), and we seek to prove that no $\{0,1\}$-solution exists. The basic DPLL algorithm works in this setting with little modification: one can still query variables and assign them to $\{0,1\}$ values; now we label leaves of the search tree with any inequality $a_i \cdot x \geq b_i$ in the system that is falsified by the sequence of assignments made on the path from the root to the leaf. If every leaf ends up being labelled with a falsified inequality, then the tree certifies that the system of inequalities has no $\{0,1\}$-solutions.

However, with the expanded domain we can consider the DPLL tree *geometrically*. To be more specific, imagine replacing each $\{0,1\}$ query to a variable x_i in the decision tree with two "inequality queries" $x_i \leq 0$ and $x_i \geq 1$. Each node u in the tree after this replacement is now naturally associated with a polyhedral set $\mathcal{P}_u$ of points satisfying each of the the input inequalities *and* each of the inequalities on the path to this node. Since we began with a DPLL refutation, it is clear that for any leaf ℓ the polyhedral set $\mathcal{P}_\ell$ associated with the leaf is empty, as any $\{0,1\}$ solution would have survived each of the inequalities queried on some path in the tree and thus would exist in one of the polyhedral sets at the leaves.

The stabbing planes system is then the natural generalization of the previous object: an SP refutation consists of a generalized DPLL refutation where each node is labelled with an *arbitrary* integral linear inequality $Ax \geq b$ (that is, the vector A and the parameter b are both integral), and the outgoing edges are labelled with the inequalities $Ax \geq b$ and $Ax \leq b - 1$. Clearly, any integral vector $x \in \mathbb{Z}^n$ will satisfy at least one of the inequalities labelling the outgoing edges, and so if the polyhedral set at each leaf (again, obtained by intersecting the original system with the inequalities on the path to the leaf) is empty then we have certified that the original system of inequalities has no integral solutions. (See Figure 1 for a simple example.) The main innovation of Stabbing Planes is its simplicity: refutations are simply decision trees that query linear inequalities. Note that the more obvious extension of DPLL to linear inequalities would branch on $Ax \geq b$ and its *actual* negation, $Ax < b$. However with this branching rule, we would have to add additional rules in order to have completeness. By branching on an inequality and its "integer negation", we are able to get by with just one rule analogous to the resolution rule in DPLL.

From the perspective of SAT solving, even though tree-like Resolution and the search for satisfying assignments encapsulated by DPLL are equivalent, it is the search point of view of DPLL that has led to the major advances in SAT algorithms now found in modern

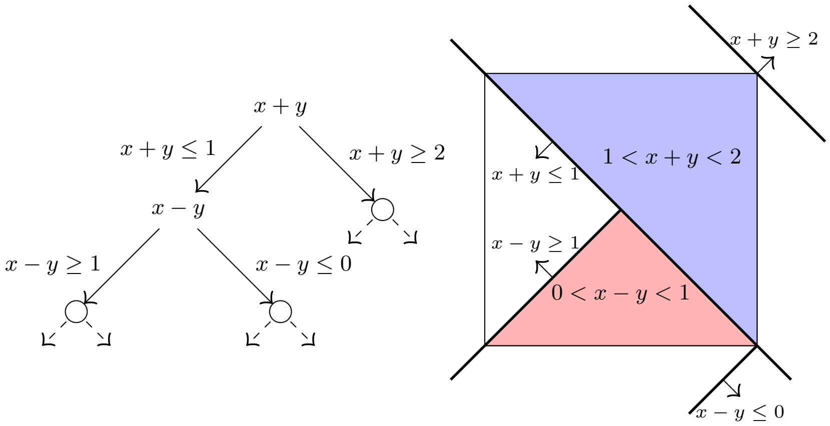

Figure 1 A partial SP refutation and the result on the unit square. The shaded areas are "removed" from the polytope, and we recurse on each side.

conflict-directed clause learning (CDCL) SAT solvers. A natural hypothesis is that it is much easier to invent useful heuristics in the language of query-based algorithms, as opposed to algorithms based on the resolution rule. Stabbing Planes offers a similar benefit with respect to reasoning about inequalities.

With the exception of mixed integer programming (MIP) solvers (such as CPLEX [19]), current solvers that, like Stabbing Planes, manipulate integer linear inequalities over Boolean variables are generally built on the same backtracking-style infrastructure as DPLL and CDCL SAT solvers but maintaining information as integer linear inequalities as opposed to clausal forms. The solvers are known as *pseudoBoolean* solvers and have been the subject of considerable effort and development.

PseudoBoolean solvers work very well at handling the kinds of symmetric counting problems associated with, for example, the pigeonhole principle (PHP), which is known to be hard for CDCL SAT solvers, as well as other problems where the input constraints are much more succinctly and naturally expressed in inequality rather than clausal form. Innovations in pseudoBoolean solvers include use of normal forms for expressing constraints, techniques to generalize fast unit propogation and watched literals from DPLL to the analogue for integer linear inequalities, as well as methods to learn from conflicts and simplify learned constraints when integer coefficients from derived inequalities get too large [29, 5]. Despite all of this, even for the best pseudoBoolean solvers, the benefits of expressibility are usually not enough compensation for the added costs of manipulating and deriving new inequalities and they outperform CDCL solvers only in very limited cases in practice [5].

A key limitation of these pseudoBoolean solvers is the fact that all branching is based on assigning values to individual variables; i.e., dividing the problem by slabs parallel to one of the coordinate axes. Stabbing Planes eliminates this constraint on the search and allows one to apply a divide and conquer search based on arbitrary integer linear constraints that are not necessarily aligned with one of these coordinate axes. This opens up the space of algorithmic ideas considerably and should allow future pseudoBoolean solvers to take fuller advantage of the expressibility of integer linear constraints. For example, a Stabbing Planes search could choose to branch on a linear inequality that is derived from the geometric properties of the rational hull of the current constraints by, say, splitting the volume, or by doing a balanced split at a polytope vertex, since properties of the rational hull can be determined efficiently. Such operations could be done in conjunction with solvers such as CPLEX to obtain the best of both kinds of approaches.

Beyond the prospect of Stabbing Planes yielding improved backtracking search for pseudoBoolean solvers, Stabbing Planes should allow the same kind of learning of inequalities from conflicts that is being done in existing pseudoBoolean solvers, and hence get the benefits of both. In this work we do not focus on the theoretical benefits of learning from conflicts because we already can show considerable theoretical benefit from the more general branching alone and because the theoretical benefits of the restricted kinds of learned linear inequalities from conflicts available even in existing solvers are not at all clear.

From a proof complexity perspective, the SP system turns out to be polynomially equivalent to the semi-algebraic proof system *tree-like* R(CP), introduced by Krajíček [23]. Roughly speaking one can think of R(CP) as a mutual generalization of both Cutting Planes and Resolution – the lines of an R(CP) proof are clauses of integer linear inequalities, and in a single step one can take two clauses and either apply a cutting-planes rule to a single inequality in each clause or apply a resolution-style “cut”. However, SP perspective turns out to be quite useful: we show that SP has quasi-polynomial size refutations of the Tseitin principle, and also that SP can polynomially-simulate Cutting Planes (neither result was previously known to hold for tree-like R(CP)).

We also investigate the relationship between SP refutations and communication complexity. Given an unsatisfiable CNF $\mathcal{F}$ and any partition of the variables (X, Y) of $\mathcal{F}$ into two sets, one can consider the following two-party search problem $\mathsf{Search}_{X,Y}(\mathcal{F})$: Alice receives an assignment to the X-variables, Bob receives an assignment to the Y-variables, and they must communicate and output a falsified clause of $\mathcal{F}$ under their joint assignment. At this time *all* strong lower bound results for Cutting Planes refutations essentially follow from studying the communication complexity of $\mathsf{Search}_{X,Y}(\mathcal{F})$. In particular, depth-$d$ (respectively, length-L tree-like, length-L space s) CP refutation of $\mathcal{F}$ yields an d-round (respectively, $O(\log L)$-round, $O(s \log L)$-round) real communication protocols for $\mathsf{Search}_{X,Y}(\mathcal{F})$, and a length-$L$ CP refutation of $\mathcal{F}$ yields a size L real communication game [24, 27, 6, 11, 16].

Each of these results has been used to derive strong lower bounds on Cutting Planes refutations by proving the corresponding lower bound against the search problem [24, 11, 17, 13, 27, 6]. Furthermore, the above lower bound techniques hold even for the stronger *semantic* Cutting Planes system (the lines of which are integer linear inequalities, and from two integer linear inequalities we are allowed to make *any* sound deduction over integer points) [12]. This makes the known lower bounds much stronger, and it is quite surprising that all one needs to exploit for strong lower bounds is that the lines are linear inequalities (rather than exploiting some weakness of the deduction rules). However, this strength also illustrates a weakness of current techniques, as once the lines of a proof system $\mathcal{P}$ become expressive enough, semantic proof techniques (i.e. ones that work for the semantic version of the proof systems) completely break down since every tautology has a short semantic proof. Therefore, it is of key importance to develop techniques which truly exploit the “syntax” of proof systems, and not just the expressive power of the lines.

Hence, it is somewhat remarkable that we are able to show that these results still hold if we replace real communication protocols with SP refutations. That is, we show:

- A depth-d CP refutation of $\mathcal{F}$ yields a depth-d SP refutation of $\mathcal{F}$.
- A length-L tree-like CP refutation of $\mathcal{F}$ yields a depth $O(\log L)$ SP refutation of $\mathcal{F}$.
- A length-L, space s CP refutation of $\mathcal{F}$ yields a depth $O(s \log L)$ SP refutation of $\mathcal{F}$.
- A length-L CP refutation of $\mathcal{F}$ yields a size $O(L)$ SP refutation of $\mathcal{F}$.

Since SP is a syntactic system this further motivates studying its depth- and size-complexity. We can use semantic techniques to get some lower bounds for SP: we show that a size-S and depth-d SP refutation yields a real communication protocol with cost $O(d)$ and for which the protocol tree has size $O(S \cdot n)$. This simulation yields new proofs of some depth lower bounds already known in the literature; however, these lower bounds are complemented by showing that neither SP refutations nor real communication protocols can be balanced. This should be viewed in a positive light: the depth- and size-complexity problems are truly different for SP, and furthermore, one seemingly cannot obtain size lower bounds for SP by proving depth lower bounds for real communication protocols (in contrast to, say, tree-like Cutting Planes). In sum, SP appears to be a very good candidate for a proof system on the "boundary" where current techniques fail to prove strong size lower bounds.

The rest of the paper is outlined as follows. After some preliminaries in Section 2, we give a simple refutation of the Tseitin problem in SP in Section 3. In Section 4, we prove a raft of simulation and equivalence results for SP – showing it is equivalent to R(CP), relating it to Cutting Planes in various measures such as depth, length, and space, and showing how an SP proof yields a real communication protocol for the canonical search problem. Finally, in Section 5, we prove depth lower bounds for SP and some impossibility results for balancing.

2 Preliminaries

Before we define the new proof system formally, we need to make a few general definitions that are relevant to semi-algebraic proof systems.

An *integer linear inequality* (or simply a *linear inequality*) in the variables $x = x_1, \ldots, x_n$ is $Ax \geq b$, where $A \in \mathbb{Z}^n$ and $b \in \mathbb{Z}$. A system of linear inequalities $\mathcal{F}$ is *unsatisfiable* if there is no Boolean assignment $\alpha \in \{0,1\}^n$ which simultaneously satisfies every inequality in $\mathcal{F}$. We sometimes refer to inequalities as lines and write $L \equiv Ax \geq b$. The integer negation of a line L is the inequality $\neg L \equiv Ax \leq b - 1$.

An unsatisfiable formula in a conjuctive normal form (CNF) defines an unsatisfiable system of linear inequalities $\mathcal{F}$ in a natural way. A clause $\bigvee_{i=1}^{k} x_i \vee \bigvee_{i=1}^{l} \neg x_i$, is translated into the inequality $\sum_{i=1}^{k} x_i + \sum_{i=1}^{l} (1 - x_i) \geq 1$, and $\mathcal{F}$ is the set of translations of all clauses. We assume that $\mathcal{F}$ always contains the axioms $x_i \geq 0$ and $-x_i \geq -1$ for all variables x_i, as we are interested in propositional proof systems for refuting unsatisfiable Boolean formulas.

▶ **Definition 1.** A *propositional proof system* $\mathcal{P}$ is a non-deterministic polynomial time Turing machine (TM) deciding the language of unsatisfiable CNF formulas. Given an unsatisfiable CNF, the NP-certificate is called *the proof* or *the refutation.*

The strength of proof systems is compared using the notion of polynomial simulation.

▶ **Definition 2.** Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two proof systems. We say that $\mathcal{P}_1$ *polynomially simulates* $\mathcal{P}_2$ if for every unsatisfiable formula $\mathcal{F}$, the shortest refutation of $\mathcal{F}$ in $\mathcal{P}_1$ is at most polynomially longer than the shortest refutation in $\mathcal{P}_2$. $\mathcal{P}_1$ is *strictly stronger* than $\mathcal{P}_2$ if $\mathcal{P}_1$ polynomially simulates $\mathcal{P}_2$, but the converse does not hold. Finally, we say that $\mathcal{P}_1$ and $\mathcal{P}_2$ are *incomparable* if neither can polynomially simulate the other.

We now describe the proof system Stabbing Planes, our central object of study.

▶ **Definition 3.** Let $\mathcal{F}$ be an unsatisfiable system of linear integral inequalities. A *Stabbing Planes (*SP*) refutation* of $\mathcal{F}$ is a threshold decision tree: a directed binary tree in which each edge is labelled with a linear integral inequality. If the right outgoing edge of a node is labelled with $Ax \geq b$, then the left outgoing edge has to be labelled with its integer negation, $Ax \leq b-1$. We refer to Ax (or the pair of inequalities $Ax \leq b - 1, Ax \geq b$) as *the query* corresponding to the node. The *slab* corresponding to the query is $\{x^* \in \mathbb{R}^n \mid b - 1 < Ax^* < b\}$.

Let the set of all paths from the root to a leaf in the tree be denoted by $\{p_1, \dots, p_\ell\}$. Each leaf i is labelled with a non-negative linear combination of inequalities in $\mathcal{F}$ with the inequalities along the path p_i that yields $0 \geq 1$.

The *size* of a SP refutation is the number of bits needed to represent every inequality in the refutation. The *length* of a SP refutation is the number of nodes in the threshold tree. The size (length) of refuting a system of linear inequalities $\mathcal{F}$ in SP is the minimum size (length) of any SP refutation of $\mathcal{F}$. The *rank* or *depth* of a SP refutation $\mathcal{P}$ is the longest root-to-leaf path in the threshold tree of $\mathcal{P}$. The rank (depth) of refuting $\mathcal{F}$ in SP is the minimum rank (depth) over all SP refutations of $\mathcal{F}$.

Refutations in SP have an intuitive geometric interpretation: each step of a refutation can be viewed as nondeterministically removing a slab from the solution space and recursing on the resulting polytopes on both sides of the slab. The aim is to recursively cover the solution space with slabs until every feasible point within this polytope is removed. An example of this can be seen in Figure 1 in the introduction. In particular, the polytope at any step of the recursion is empty if and only if there exists a convex combination of the axioms and inequalities labelling the corresponding root-to-leaf path in the refutation equivalent to $0 \geq 1$. This is summarized in the following fact which follows directly from the Farkas' lemma. The "moreover" part of the following fact is an application of Carathéodory's theorem, and will be useful for technical reasons later in the paper. We refer the interested reader to [30] for some background on polytope theory.

▶ **Fact 4.** *Let $\mathcal{F} = \{A_1 x \geq b_1, \dots, A_m x \geq b_m\}$ be a system of integer linear inequalities. The polytope defined by $\mathcal{F}$ is empty if and only if there is a non-negative (rational) linear combination of the inequalities of $\mathcal{F}$ which evaluates to $0 \geq 1$. Moreover, the non-negative linear combination can be taken to be supported on $\leq n$ of the inequalities from $\mathcal{F}$, where n is the dimension of the space to which x belongs.*

It is straightforward to see that SP is a sound and complete proof system. Completeness follows from a simple observation that SP polynomially simulates DPLL. To see that SP is sound, let $\mathcal{R}$ be a SP refutation of some formula $\mathcal{F}$. Observe that for any node in $\mathcal{R}$ with outgoing edges labelled $Ax \geq b$ and $Ax \leq b - 1$, any $0-1$ assignment to the variables $\alpha \in \{0,1\}^n$ must satisfy exactly one of the two inequalities. Therefore, if a Boolean solution $\alpha \in \{0,1\}^n$ satisfies $\mathcal{F}$, then for at least one of the leaves of $\mathcal{R}$, one cannot derive $0 \geq 1$. This follows by Fact 4 because the polytope formed by the inequalities labelling root-to-leaf path is non-empty (α lies in this polytope) .

Next we recall a well-known and extensively-studied proof system: Cutting Planes (CP). For an introduction to Cutting Planes, we refer an interested reader to Chapter 19 in [21].

▶ **Definition 5.** Let $\mathcal{F}$ be an unsatisfiable system of linear inequalities. A *Cutting Planes* (CP) *refutation* of $\mathcal{F}$ is a sequence of linear inequalities $\{L_1, \dots, L_\ell\}$ such that $L_\ell = 0 \geq 1$ and each L_i is either an axiom $\in \mathcal{F}$ or is obtained from previous lines via one of the following inference rules. Let α, β be positive integers.

$$\text{Linear Combination: } \frac{Ax \geq a \qquad Bx \geq b}{(\alpha A + \beta B)x \geq \alpha a + \beta b} \qquad \text{Division: } \frac{\alpha A x \geq b}{Ax \geq \lceil \frac{b}{\alpha} \rceil}$$

We refer to ℓ as the *length* of the refutation. The *length* of refuting $\mathcal{F}$ in CP is the minimum length of a CP refutation of $\mathcal{F}$.

The directed acyclic graph (DAG) $G = (V, E)$ associated with a CP refutation $\{L_1, \dots, L_\ell\}$ is defined as follows. We have $V = \{L_1, \dots, L_\ell\}$ and $(u, v) \in E$ if and only if the line labelling v was derived by an application of an inference rule involving the line labelling u. Without loss of generality, we may assume that there is only one vertex with out degree 0, which we call the root. The root of G is labelled with L_ℓ and the leaves are labelled with the axioms.

The *rank* or *depth* of the refutation is the length of the longest root-to-leaf path in G. The rank of refuting an unsatisfiable system of linear inequalities $\mathcal{F}$ is the minimum rank of any refutation of $\mathcal{F}$ in the given proof system. Finally, *tree-like* CP is defined by restricting proofs to be such that the underlying graph G is a tree.

It is not known if an arbitrary SP refutation can be transformed into a refutation with coefficients of polynomial bitsize. Therefore we currently must make the distinction between SP refutation size and length. Fortunately, all of our results hold in the best possible scenario; our upper bounds are low weight (polynomial-length); the simulations are length preserving, and our lower bounds hold for any weight.

3 Motivating Example: SP Refutations of Tseitin Formulas

Tseitin contradictions are among the most well-studied unsatisfiable formulas in proof complexity, and are the quintessential formulas that are believed to be hard for CP [21]. Despite the fact that exponential lower bounds for CP are known for many natural families of formulas (including recent lower bounds for random $O(\log n)$-CNF formulas), there are no nontrivial lower bounds known for the Tseitin contradictions, and for good reason: the only known lower bound method for CP reduces the problem of refuting a formula in CP to a monotone circuit problem, for which the corresponding monotone circuit problem for Tseitin contradictions is easy.

In this section, we demonstrate the power of Stabbing Planes by showing that there exists a shallow quasi-polynomial size SP refutation of the Tseitin formulas. This, together with our simulation results from Section 4, show that SP is provably more powerful than CP in terms of depth, and strongly suggests that SP is strictly more powerful than CP.

Tseitin contradictions are any unsatisfiable family of mod-2 equations subject to the constraint that every variable occurs in exactly two equations. An instance of Tseitin, denoted $\mathrm{Tseitin}(G, \ell)$ is defined by a connected undirected graph $G = (V, E)$ and a node labelling $\ell \in \{0,1\}^V$ of odd total weight: $\sum_{v \in V} \ell_v = 1 \bmod 2$. For each edge $e \in E$ there is a variable x_e in $\mathrm{Tseitin}(G, \ell)$, and for each vertex $v \in V$ an equation $\sum_{e \ni v} x_e \equiv \ell_v \mod 2$, stating that the sum of the variables x_e incident with v is ℓ_v mod 2. The edge equations sum to zero mod 2 since every variable occurs exactly twice, but the vertex equations sum to one mod 2, since the node labelling is odd, and therefore the equations are unsatisfiable. When G has degree D, we can express $\mathrm{Tseitin}(G, \ell)$ as a D-CNF formula containing $|V| \cdot 2^{D-1}$ clauses.

The obvious way to refute $\mathrm{Tseitin}(G, \ell)$ under an assignment x is to find a vertex w for which the corresponding vertex equation is falsified. This can be achieved by the following divide-and-conquer procedure, which maintains a set $U \subseteq V$ such that $w \in U$. The process begins by setting $U = V$. Then, V is partitioned arbitrarily into two sets V_1, V_2 of roughly the same size. Query x_e for all edges e crossing the cut (V_1, V_2), and suppose that the sum of all such x_e is odd (the case when it is even is similar). We know that either $\sum_{v \in V_1} \ell_v$ or $\sum_{v \in V_2} \ell_v$ is even: if the first sum is even then the Tseitin formula restricted to V_1 contains a contradiction, and otherwise the formula restricted to V_2 contains a contradiction. In either case, we can remove roughly half of the graph and recurse.

By keeping track of a few more variables, we can repeat this procedure recursively until $|U| = 1$. Since we reduce the size of U by half each time, this procedure results in the recursion depth logarithmic in $|V|$. It turns out that this procedure can be realized in Stabbing Planes, where recursion depth roughly corresponds to the depth of the refutation. This results in a quasi-polynomial size refutation.

▶ **Theorem 6.** *Let $G = (V, E)$ be an undirected graph, and let ℓ be a $\{0, 1\}$ vertex labelling with odd total weight. Then* $\mathrm{Tseitin}(G, \ell)$ *has an* SP *refutation of size $n^{O(\log n + D/\log n)}$ and rank $O(D + \log^2 n)$, where $n = |V|$ and D is the maximum degree in G.*

Proof. If $U \subseteq V$ is a set of vertices, then let $E(U) = \{uv \in E \mid u, v \in U\}$, and $\mathrm{Cut}(U) = \{uv \in E \mid u \in U, v \in \overline{U}\}$. Similarly, if $U_1, U_2 \subseteq V$ are disjoint then we let $\mathrm{Cut}(U_1, U_2) = \{uv \in E \mid u \in U_1, v \in U_2\}$. We construct the SP refutation recursively. During the recursion we maintain a set U of current vertices (initially $U = V$). At each recursive step, we split U into two halves U_1 and U_2 and query the total weight k of the edges crossing (U_1, U_2) via SP inequalities. Knowing k, a few additional queries allows us to determine which of U_1 or U_2 contains a contradiction, and we then recurse on the corresponding set of vertices.

We construct a proof while maintaining the following invariant: for the current subset of vertices $U \subseteq V$, we have queried linear inequalities implying that $\sum_{e \in \mathrm{Cut}(U)} x_e = k$ for some $0 \leq k \leq |\mathrm{Cut}(U)|$ such that $k \not\equiv \sum_{v \in U} \ell_v \bmod 2$. Note that this invariant ensures that our Tseitin instance restricted to the edges incident on U is unsatisfiable, since summing up all vertex constraints within U yields $\sum_{e \in \mathrm{Cut}(U)} x_e + \sum_{e \in E(U)} 2x_e \equiv k \not\equiv \sum_{v \in U} \ell_v \pmod 2$.

Initially we have $U = V$ and the invariant clearly holds. Now, let U be the current set of vertices. By the invariant we know that $\sum_{e \in \mathrm{Cut}(U)} x_e = k$ for some $k \not\equiv \sum_{v \in U} \ell_v \bmod 2$. Partition U into two halves U_1 and U_2 arbitrarily, subject to $|U_1| = \lfloor |U|/2 \rfloor$. We first determine the value of the edges going between U_1 and U_2 by querying $\sum_{e \in \mathrm{Cut}(U_1, U_2)} x_e \geq \beta$ for $\beta = 1, \ldots, |\mathrm{Cut}(U_1, U_2)|$. To each leaf of this tree we attach a second binary search tree for determining the value $|\mathrm{Cut}(U_1)|$ by querying $\sum_{e \in \mathrm{Cut}(U_1)} x_e \geq \gamma$ for $\gamma = 1, \ldots, |\mathrm{Cut}(U_1)|$. After these queries, at each leaf of the "combined" tree we will have $\sum_{e \in \mathrm{Cut}(U_1, U_2)} x_e = \beta$ and $\sum_{e \in \mathrm{Cut}(U_1)} x_e = \gamma$ for some β and γ. Furthermore, since $|\mathrm{Cut}(U_1)| + |\mathrm{Cut}(U_2)| = |\mathrm{Cut}(U)| + 2|\mathrm{Cut}(U_1, U_2)|$, we will have $\sum_{e \in \mathrm{Cut}(U_2)} x_e = \delta$, where $\delta + \gamma = k + 2\beta$, $0 \leq \delta \leq |\mathrm{Cut}(U_2)|$.

For any leaf of this tree where $\delta > |\mathrm{Cut}(U_2)|$, we can derive a contradiction by summing the axioms $-x_e \geq -1$ for all $e \in \mathrm{Cut}(U_2)$ with $\sum_{e \in \mathrm{Cut}(U_2)} x_e \geq \delta$. Otherwise, for the remaining leaves observe that $\delta + \gamma \equiv k \not\equiv \sum_{v \in U_1} \ell_v + \sum_{v \in U_2} \ell_v \pmod 2$. Now, if $\gamma \not\equiv \sum_{v \in U_1} \ell(v) \bmod 2$, then recurse on U_1. Otherwise, $\delta \not\equiv \sum_{v \in U_2} \ell(v) \bmod 2$. Then recurse on U_2.

Our recursion terminates when U contains a single vertex v. By the invariant, we have derived $\sum_{e \in \mathrm{Cut}(v)} x_e \equiv k \not\equiv \ell(v) \bmod 2$ for some $0 \leq k \leq |\mathrm{Cut}(\{v\})|$. The axioms of $\mathrm{Tseitin}(G, \ell)$ rule out Boolean assignments to the variables x_e for $e \in \mathrm{Cut}(\{v\})$, which contradict $\ell(v)$; these axioms do not prohibit incorrect fractional assignments. Therefore, to derive a contradiction, we still need to enforce that the variables x_e for $e \in \mathrm{Cut}(\{v\})$ take $\{0, 1\}$ values. We achieve this by querying all variables x_e for $e \in \mathrm{Cut}(\{v\})$ via SP inequalities $x_e \geq 1, x_e \leq 0$. This results in a complete binary tree of depth $\leq D$. Clearly, $0 \geq 1$ is immediately obtained at the leaves that disagree with $\sum_{e \in \mathrm{Cut}(\{v\})} x_e = k$. At the leaves that agree with $\sum_{e \in \mathrm{Cut}(\{v\})} x_e = k$, the inequality $0 \geq 1$ immediately follows from the assignment to the edges incident to v and one of the axioms of Tseitin.

Finally, we analyze the size and rank of the constructed SP proof. In each recursive step, we make $O((nD)^2)$ queries to determine weights of edges crossing the two cuts. Each recursive step is computed by a pair of binary trees, each of depth at most $\log(nD)$. Our recursion terminates in $\log n$ rounds because we halve the number of current vertices in each step. Once the recursion terminates, we query the variables corresponding to edges incident to the single remaining vertex – this contributes 2^D factor to size and increases depth by at most D. Overall, the SP proof has size $n^{O(\log n + D/\log n)}$ and rank $O(D + \log^2 n)$. ◀

▶ **Corollary 7.** SP *is strictly stronger than* CP *with respect to proof rank.*

Proof. By Theorem 11, any Cutting Planes refutation of rank r can be converted into a SP refutation of rank $O(r)$. Buresh-Oppenheim et al. proved $\Omega(n)$ lower bound on the rank of Cutting Planes of the Tseitin formulas on constant-degree expander graphs [7], while Theorem 6 shows that SP can refute such Tseitin formulas in rank $O(\log^2 n)$. ◀

4 Simulation Theorems

In this section, we prove simulation theorems relating the SP proof system to other similar proof systems in the literature. We begin by showing that SP is polynomially equivalent to the tree-like R(CP) system (introduced by Krajicek in [23]), which can be thought of as tree-like Resolution with clauses of inequalities and allowing CP rules. Since tree-like R(CP) simulates tree-like CP, the natural question is whether SP (and consequently R(CP)) can simulate general CP. We answer this question positively by providing two simulations. First of all, we observe that SP can depth-simulate CP. This simulation, while preserving depth of the proof, can lead to an exponential increase in the size. Via a different simulation we show that SP can size-simulate CP. This time around, while the simulation preserves the size of a CP refutation, it can significantly increase the depth. It is an interesting open question whether there is a simulation of CP by SP that can simultaneously preserve depth and size of CP refutations.

To complete the picture, we note that general R(CP) can trivially simulate tree-like R(CP) (and consequently SP) and CP. We also show that tree-like CP refutations can be efficiently converted into balanced (logarithmic-depth) SP refutations – this shows that tree-like CP refutations, which cannot in general be balanced, *can* be balanced in SP.

$$\begin{tikzcd}
 & \mathsf{R(CP)} & & \\
\mathsf{CP} \arrow[ur] \arrow[r] & \mathsf{SP} \arrow[r, phantom, "="] & \mathsf{Tree\text{-}R(CP)} \arrow[ul] \\
 & \mathsf{Tree\text{-}CP} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

We then turn to the question of space-time simulations. Recall, that a proof system can be thought of as a non-deterministic Turing machine. The notion of space of CP refutations intuitively corresponds to the minimum size of the work tape of such a non-deterministic Turing machine that is required to carry out the computation. In this analogy, the notion of length of CP refutations corresponds to the running time of the given TM. We show that general CP refutations that use length ℓ and space s can be turned into depth $O(s \log \ell)$ SP refutations. Thus, sufficiently strong lower bounds on the depth of SP refutations lead to time-space tradeoffs for CP.

4.1 Equivalence of SP with tree-like RCP

We show the SP system is polynomially equivalent to the R(CP) proof system. R(CP) proof system is formally defined as follows.

▶ **Definition 8.** The R(CP) proof system is a syntactic proof system defined as follows. The lines of the R(CP) system are disjunctions of integer linear inequalities $\Gamma = L_1 \vee L_2 \vee \cdots \vee L_\ell$, and the lines are equipped with the following deductive rules. Let Γ be an arbitrary disjunction of integer linear inequalities, let $Ax \geq b, Cx \geq d$ be arbitrary integer linear inequalities, and let α, β be any positive integers.

$$\text{Linear Combination: } \frac{(Ax \geq b) \vee \Gamma \qquad (Cx \geq d) \vee \Gamma}{(\alpha A + \beta C)x \geq (\alpha b + \beta d) \vee \Gamma} \qquad \text{Division: } \frac{(\alpha Ax \geq b) \vee \Gamma}{(Ax \geq \lceil b/\alpha \rceil) \vee \Gamma}$$

$$\text{Axiom Introduction: } \frac{}{(Ax \geq b) \vee (Ax \leq b-1)} \qquad \text{Weakening: } \frac{\Gamma}{(Ax \geq b) \vee \Gamma}$$

$$\text{Cut: } \frac{(Ax \geq b) \vee \Gamma \qquad (Ax \leq b-1) \vee \Gamma}{\Gamma} \qquad \text{Elimination: } \frac{(0 \geq 1) \vee \Gamma}{\Gamma}$$

An R(CP) proof is *tree-like* if the proof DAG is a tree.

The following two theorems state the polynomial equivalence between SP and R(CP). Due to space considerations, we refer the reader to the full version [1] of this paper for the proofs of the two theorems.

▶ **Theorem 9.** *Let $\mathcal{C}$ be an unsatisfiable CNF, and let $C_1, C_2, \ldots, C_m$ be the representation of $\mathcal{C}$ as an integer-linear system of inequalities. For any* SP *refutation of $\mathcal{C}$ with size s and depth d there is a tree-like* R(CP) *refutation of $\mathcal{C}$ of size $O(s(d^2 + dm))$ and width $d + 1$.*

▶ **Theorem 10.** *Let $\mathcal{C}$ be an unsatisfiable CNF, and let $C_1, C_2, \ldots, C_m$ be the representation of $\mathcal{C}$ as an integer linear system of equations. For any tree-like* R(CP) *proof of the disjunction $\neg C_1 \vee \neg C_2 \vee \cdots \vee \neg C_m$ with size s and depth d there is an* SP *refutation of $\mathcal{C}$ of size at most $2s$ and rank at most $2d$.*

4.2 SP simulations of Cutting Planes

SP simulates CP in two ways: via a depth-preserving simulation and via a size-preserving simulation. It is an interesting open problem if both depth and size can be preserved simultaneously during a simulation. Due to space requirements, we simply state the depth-preserving simulation and leave its proof to the full version of the paper [1].

▶ **Theorem 11.** *For every Cutting Planes refutation of rank d, there is a* SP *refutation of the same tautology with rank at most $2d$. Moreover, if the* CP *refutation is tree-like of size s then the resulting* SP *refutation is of size $O(s)$ and rank $2d$.*

Next, we present the size-preserving simulation with full details.

▶ **Theorem 12.** SP *polynomially simulates* CP.

Proof. Let $\mathcal{R} = \{A_1x \geq a_1, A_2x \geq a_2, \ldots, A_mx \geq a_m\}$ be a CP refutation of an unsatisfiable set of integer linear inequalities $\mathcal{F}$. We construct a SP refutation of $\mathcal{F}$ line by line, following $\mathcal{R}$. Our SP refutation is a tree where the right-most path is of length $m+1$ with edges labelled $A_1 \geq b_1, \ldots, A_m \geq b_m$. The left child of node $i \leq m$ along this path is labelled with $0 \geq 1$, which is derived as a non-negative linear combination of $A_jx \geq a_j$ for $j < i$, $A_ix \leq a_i - 1$, and $\mathcal{F}$. The last node in the path is also labelled with $0 \geq 1$. Since $A_mx \geq a_m \equiv 0 \geq 1$, the last node is trivially labelled with $0 \geq 1$. Thus, we only need to show that the left child of every node can be legally labelled with $0 \geq 1$. If $A_ix \geq a_i$ is an axiom, we can derive $0 \geq 1$ by subtracting $A_ix \leq a_i - 1$ from $A_ix \geq a_i \in \mathcal{F}$. If $A_ix \geq a_i$ is a non-negative combination of two previous inequalities, i.e., $A_ix \geq a_i$ is $\alpha A_{j_1}x + \beta A_{j_2}x \geq \alpha a_{j_1} + \beta a_{j_2}$ for some $j_1, j_2 < i$ and $\alpha, \beta \in \mathbb{Z}_{\geq 0}$, we can derive $0 \geq 1$ by subtracting $A_ix \leq a_i - 1$ from the non-negative linear combination of $A_{j_1}x \geq a_{j_1}$ and $A_{j_2}x \geq a_{j_2}$ used to derive $A_ix \geq a_i$. Finally, suppose that $A_ix \geq a_i$ is obtained by an application of the division rule to $A_jx \geq a_j$ for some $j < i$, i.e., $A_ix \geq a_i$ is $\frac{A_j}{c}x \geq \left\lceil \frac{a_j}{c} \right\rceil$ where $c \in \mathbb{N}$ divides every entry in A_j. On the path to this node in our SP refutation we queried $A_jx \geq a_j$. Dividing this inequality by c and subtracting $A_ix \leq a_i - 1$, we obtain $0 \geq a_j/c - (\lceil a_j/c \rceil - 1)$. This gives us $0 \geq \frac{a_j}{c} + 1 - \left\lceil \frac{a_j}{c} \right\rceil$. Since the right-hand side is strictly positive this can be normalized to give $0 \geq 1$. ◀

4.3 CP and Balanced SP

It is known that CP refutations cannot be balanced (i.e. size-s refutations being turned into size $O(s)$ depth $O(\log s)$ refutation) in CP. Here, we show that CP proofs can be turned into balanced SP proofs. More specifically, we prove the following.

▶ **Theorem 13.** *Suppose there is a size s tree-like* CP *refutation of a set of linear integer inequalities $\mathcal{F}$. Then there is a size s depth $O(\log s)$* SP *refutation of $\mathcal{F}$.*

Proof. The construction is recursive. Let T be the tree corresponding to $\mathcal{P}$. If $|T| = O(1)$, use one of the previous simulation theorems to create an SP refutation of $\mathcal{P}$. Now, let v be a node in T such that the subtree T_v rooted at v satisfies $|T|/3 \leq |T_v| \leq 2|T|/3$, which must exist since the size measure is additive. Let $Bx \geq b$ be the line in $\mathcal{P}$ corresponding to v. Our SP simulation starts by querying Bx. If $Bx \geq b$ then we apply the recursive construction to $T \setminus T_v$, treating $Bx \geq b$ as a new axiom. Otherwise, if $Bx \leq b - 1$ (which contradicts the the input set of inequalities $\mathcal{F}$) we apply the recursive construction to T_v. The size is clearly preserved, and the depth of the proof becomes logarithmic, since we are reducing the size of the proof to be simulated by a constant factor on each branch of a query. ◀

We can also show that bounded depth and space CP refutations yield balanced SP proofs. See full version [1] for the proof of the following theorem.

▶ **Theorem 14.** *Suppose that we have a length ℓ, space s* CP *refutation of an unsatisfiable set of linear integral inequalities. Then there is a depth $O(s \log \ell)$* SP *refutation of the same set of linear integral inequalities.*

5 Impossibility Results

In this section, we prove near-optimal lower bounds on SP rank via reductions to randomized and real communication complexity. We then tackle the harder problem of proving unrestricted superpolynomial size lower bounds for SP. Although we are unable to prove such lower bounds we explain why current approaches fail. Essentially all lower bounds for CP have been obtained by reducing to a communication complexity problem; in the case of tree-like CP, the reduction is to the communication complexity of a corresponding search problem. For more general dag-like CP, the reduction is to the size of "communication games" [13, 17] (communication games are a dag-like model of communication that gives an equivalence between communication *size* and monotone circuit size, analogous to the famous equivalence between communication *depth* and monotone formula size). Although tree-like CP proofs cannot be balanced in general, communication protocols (both deterministic and randomized) can be balanced, and thus tree-like CP lower bounds follow from communication complexity lower bounds. Similarly, we show that it is not possible to balance SP refutations, and thus we cannot in general obtain size lower bounds directly from rank lower bounds. Moreover, we show that SP refutations imply real communication protocols, and unlike ordinary communication protocols, we show that real protocols *cannot* in general be balanced. This rules out proving length lower bounds on SP refutations from (real) communication complexity lower bounds.

5.1 SP Refutations Imply Communication Protocols

Real communication protocols were introduced by Krajíček [24]. In this model, the players are allowed to communicate by sending real-valued functions of their inputs to a referee who announces their comparison.

▶ **Definition 15.** A *real communication protocol* is a full binary tree in which every non-leaf node v is labeled with a pair of functions $a^v : \mathcal{X} \to \mathbb{R}, b^v : \mathcal{Y} \to \mathbb{R}$, and the leaves are labelled with elements in $\mathcal{Z}$. Two players, Alice and Bob, receive inputs from $\mathcal{X} \times \mathcal{Y}$, with Alice receiving $x \in \mathcal{X}$ and Bob receiving $y \in \mathcal{Y}$. Beginning at the root, the players traverse the tree as follows: at each node, they send real values $a^v(x)$ and $b^v(y)$ to a "referee" who returns (to both of them) a bit indicating the result of the comparison $a^v(x) \geq b^v(y)$; the players proceed to the left child if $a^v \geq b^v$, and to the right child otherwise. Once they reach a leaf, the protocol halts, and the players output the value in $\mathcal{Z}$ labelling the leaf; it computes a function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ in the natural way.

The *cost* of a real protocol is the depth of the tree, or equivalently the maximum number of rounds of communications with the referee over any input (x, y), and the *size* is the number of nodes in the protocol. Similarly, the cost (size) of computing a function f is the smallest cost (size) real protocol computing f.

Krajíček showed that from a low-rank CP refutation of an unsatisfiable CNF, one can obtain a real communication protocol for solving a related search problem [24]. We describe this search problem next.

▶ **Definition 16.** Let $\mathcal{F} = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an unsatisfiable CNF and (X, Y) be a partition of the variables. The relation $\mathsf{Search}_{X,Y}(\mathcal{F}) \subseteq \{0,1\}^X \times \{0,1\}^Y \times [m]$ consists of all triples (x, y, i) such that the total assignment $z = (x, y)$ to all of the variables of C_i falsifies the clause i.

The search problem is the natural interpretation of a refutation in the setting of communication. Indeed, essentially every lower bound for CP has been proved by reducing to the communication complexity of the search problem. In a similar manner, we show that SP refutations can be turned into both randomized and real protocols for the search problem which preserve the rank of the refutation.

▶ **Lemma 17.** *Let $\mathcal{F}$ be an unsatisfiable CNF formula and (X, Y) be any partition of the variables. Any* SP *refutation of $\mathcal{F}$ of rank r implies a real communication protocol of cost $O(r + \log n)$ and an $O(r \log n + \log^2 n)$ randomized bounded-error protocol for* $\mathsf{Search}_{X,Y}(\mathcal{F})$.

The protocol consists of traversing the SP tree until a leaf is reached (r rounds), then finding a clause falsified at the leaf in $O(\log n)$ rounds: note that evaluating any linear inequality can be done using a single bit of communication.

For the second part of the lemma, Alice and Bob run the ε-error $O(\log m + \log \varepsilon^{-1})$-bit protocol of Nisan [26] for deciding an m-bit linear inequality. By the well-known result of Muroga [25], any inequality on n Boolean variables only requires coefficients representable by $O(n \log n)$ bits (recall that Alice and Bob's input will always be a Boolean assignment and so this suffices). Because there are at most $r + \log n$ inequalities evaluated along any root-to-leaf path in the refutation, the protocol is repeated at most $r + \log n$ many times. By a union bound, we require $\varepsilon < c/(\log n + r)$, where c is some constant bound on the error that we allow. Therefore, every inequality can be computed in $O(\log n + \log r)$ many bits to compute, giving a $O(r \log n + \log^2 n)$ bounded-error randomized protocol for $\mathsf{Search}_{X,Y}(\mathcal{F})$.

5.2 Rank Lower Bounds For SP

To take advantage of Lemma 17, we need to find some candidate formulas on which to prove rank lower bounds and then study the search problem obtained from applying this transformation. We do so for both the Tseitin formulas and a variant of the pebbling contradictions, a reformulation of the classical black pebbling games as an unsatisfiable 3-CNF formula, originally introduced by Ben-Sasson et al. [4, 3].

The black pebbling game can be phrased as a contradictory 3-CNF as follows: Let G be a DAG with a set of source nodes $S \subseteq V(G)$ (having fanin 0), a unique sink node t (with fanin 2 and fanout 0), and the remaining nodes each having fanin exactly 2. The pebbling contradictions Peb_G consists of the following $n+1$ clauses over variables $v \in V(G)$: sink axiom: a single clause $\neg t$; source axioms: a clause s for every source $s \in S$; pebbling axioms: a clause $\neg u \vee \neg v \vee w$ for every $w \in V \setminus S$ with immediate children u, v.

Unfortunately, both the pebbling contradictions and the Tseitin formulas have short SP refutations. In particular, for any graph G, the polytope formed by the constraints of Peb_G is empty and therefore a nonnegative combination of the constraints yielding $0 \geq 1$ exists, this is a valid rank-1 SP refutation. For Tseitin, this follows from the poly-logarithmic rank upper bound in Theorem 6. We modify these formulas to make them harder to solve.

A standard technique for amplifying the hardness of computing some function $f : \mathcal{X}^n \rightarrow \mathcal{Z}$ is by *lifting* that function. This is done by obscuring the input variables by replacing them each by a small function $g : \mathcal{A} \rightarrow \mathcal{X}$ known as a *gadget*, which must be evaluated before learning the input to the original function. For an input $\alpha \in \mathcal{A}^n$, the function f *lifted* by gadget g is then $(f \circ g^n)(\alpha) = (g(\alpha_1), \ldots, g(\alpha_n))$. The intuition is that this lifted function $f \circ g^n$ should be much harder than the original because the players must first evaluate the gadget $g(\alpha_i)$ to learn each bit of the actual input to the function f. Furthermore, intuition says that if the gadget is sufficiently difficult to compute, then the model will be reduced to using much more rudimentary methods to evaluate the lifted function.

The standard hard-to-compute gadget is the *pointer* or *index* gadget, $\mathsf{IND}_\ell : [\ell] \times \{0,1\}^\ell \rightarrow \{0,1\}$. For an input $(x, y) \in [\ell] \times \{0,1\}^\ell$, x is a $\log \ell$-bit string encoding a pointer into the ℓ-bit string $y \in \{0,1\}^\ell$. The output of $\mathsf{IND}_\ell(x, y)$ is $y[x]$, the x-th bit in the string y. This is most often applied in communication complexity, where typically the variable partition between the players is that Alice is given $x \in [\ell]$ and Bob is given $y \in \{0,1\}^\ell$. In any standard model of communication, for this partition of the variables, it is difficult to imagine any communication protocol which could compute the index gadget with significant advantage over the trivial protocol; sending every bit of Alice's pointer x to Bob.

Raz and McKenzie formalized this intuition, in what has become known as a lifting theorem [28, 15]. They show that deterministic communication protocols cannot compute any function f lifted by the index gadget significantly better than simply mimicking a decision tree computing f, and performing the trivial protocol for evaluating the index gadget every time a bit of the input to f is needed.

Lifting theorems for real communication were originally proved by Bonet et al. [6] based on the techniques of Raz-McKenzie. Their theorem lifts lower bounds on the decision tree complexity of a function f to lower bounds on the cost of real communication protocols computing $f \circ \mathsf{IND}_\ell^n$. The decision tree complexity $\mathsf{DT}(f)$ of a function f is simply the minimum depth need by any decision tree to compute f. We use a simplified lifting theorem for real communication by de Rezende et al. [11], which we state next.

▶ **Theorem 18.** *(de Rezende et al. [11]) Let f be a function with domain $\{0,1\}^n$ and let $\ell = n^4$. If there is a real communication protocol of cost c that solves $f \circ \mathsf{IND}_\ell^n$ where Alice is given $x \in [\ell]^n$ and Bob is given $y \in \{0,1\}^{n\ell}$, then there is a decision tree solving f using $O(c/\log \ell)$ queries.*

Our goal is now to is to combine this theorem with Lemma 17 in order to prove lower bounds on the rank of SP refutations of $\mathrm{Peb}_G \circ \mathsf{IND}_\ell^n$. Syntactically speaking though,

$\text{Peb}_G \circ \mathsf{IND}_\ell^n$ is not a valid input to our proof system. Therefore, we must show that the lifted function can indeed be phrased as a small CNF formula. The following encoding is due to Beame et al. [2]:

Let $\mathcal{F} = C_1 \vee \ldots \vee C_m$ be a CNF formula over variables $x_1, \ldots, x_n$. The CNF representing $\mathcal{F} \circ \mathsf{IND}_\ell^n$ is defined on new sets of variables $y_{i,j}$ and $z_{i,j}$ for all $i \in [n]$ and $j \in [\ell]$. This CNF has the following set of clauses

- Pointer clauses: for each $i \in [n]$, a clause $y_{i,1} \vee \ldots \vee y_{i,\ell}$.
- $\mathcal{F}$-clauses: for each clause $C_i \in \mathcal{F}$, where $C_i = y_{i_1} \vee \ldots \vee y_{i_k} \vee \neg x_{i_{k+1}} \vee \ldots \vee \neg x_{i_s}$ and for every $(j_1, \ldots, j_n) \in [\ell]^n$, a clause $(y_{i_1,j_1} \to z_{i_1,j_1}) \vee \ldots \vee (y_{i_k,j_k} \to z_{i_k,j_k}) \vee (y_{i_{k+1},j_{k+1}} \to \neg z_{i_{k+1},j_{k+1}}) \vee \ldots \vee (y_{i_s,j_s} \to \neg z_{i_s,j_s})$.

We will abuse notation and use $\mathcal{F} \circ \mathsf{IND}_\ell^n$ to denote the function, as well as it's CNF formulation, and use context to differentiate between the two.

A final subtlety that should be mentioned is that applying Theorem 18 to an SP refutation of $\text{Peb}_G \circ \mathsf{IND}_\ell^n$ yields a protocol for $\mathsf{Search}_{X,Y}(\text{Peb}_G \circ \mathsf{IND}_\ell^n)$ which is not in the correct form to apply Theorem 18 ($\mathsf{Search}_{X,Y}(\text{Peb}_G \circ \mathsf{IND}_\ell^n)$ is a function of a lifted function, whereas Theorem 18 can only be applied to lifted functions). Luckily, this is not a significant issue; Huynh et al. [18] show that, for any unsatisfiable CNF $\mathcal{F}$, any real communication protocol for $\mathsf{Search}_{X,Y}(\mathcal{F} \circ \mathsf{IND}_\ell^n)$, where $X = [\ell]^n$ and $Y = \{0,1\}^{n\ell}$, implies a real communication protocol for $\mathsf{Search}_{X,Y}(\mathcal{F}) \circ \mathsf{IND}_\ell^n$ with the same parameters.

It is now straightforward to obtain lower bounds on the rank of SP refutations. For the lifted pebbling formulas, SP rank lower bounds follow from combining Lemma 17 and Theorem 18 with a lower bound on the complexity of decision trees solving Peb_G proved by de Rezende et al.[11].

▶ **Theorem 19.** *There exists a graph G of indegree 2 on n vertices such that the unsatisfiable CNF formula $\text{Peb}_G \circ \mathsf{IND}_\ell^n$, for $\ell = n^4$, on $n(\ell + \log \ell)$ variables requires rank $\Omega(\sqrt{n \log n})$ to refute in* SP.

Proof. de Rezende et al. [11] showed the existence of a graph G on n vertices with indegree 2 such that the decision tree complexity of outputting a falsified clause of Peb_G is $\Omega(\sqrt{n/\log n})$. Applying Theorem 18 and combining this with the fact that shallow SP refutations give efficient protocols for the associated search problem (Lemma 17), proves the desired $\Omega(\sqrt{n \log n})$ lower bound on the rank of SP refutations of $\text{Peb}_G \circ \mathsf{IND}_\ell^n$. ◀

Finally, a similar technique can be applied to obtain a lower bound on the rank of SP refutations for a lifted variant of the Tseitin formulas. This follows from the lower bound on the randomized communication complexity of the search problem for the Tseitin formulas lifted by a small constant-size gadget, which was obtained by Göös and Pitassi [14]. In particular, they use the *versatile gadget*, $\mathsf{VER} : \mathbb{Z}_4 \times \mathbb{Z}_4 \to \{0,1\}$, which is defines as $\mathsf{VER}(x,y) = 1 \iff x + y \pmod 4 \in \{2,3\}$.

▶ **Theorem 20.** *(Göös and Pitassi [14]) There exists a constant-degree graph G on n vertices such that, if ℓ is any $\{0,1\}$ vertex labelling with odd total weight and (X,Y) is any partition of the variables, any bounded-error randomized communication protocol for $\mathsf{Search}_{X,Y}(\text{Tseitin}(G,\ell) \circ \mathsf{VER}^n)$ on $O(n)$ variables, requires $\Omega(n/\log n)$ bits of communication.*

Furthermore, Göös and Pitassi showed how, for any CNF formula $\mathcal{F}$, the composed function $\mathcal{F} \circ \mathsf{VER}^n$ can be encoded as a CNF formula. For brevity, we refer the reader to Göös and Pitassi [14] for the definition of encoding of $\mathcal{F} \cdot \mathsf{VER}^n$ as a CNF formula, and recall that if $\mathcal{F}$ is a CNF with m clauses and width w, then $\mathcal{F} \circ \mathsf{VER}^n$ contains at most $m \cdot 2^{4w}$

clauses. The width of every clause in the Tseitin formulas are bounded by maximal degree d in the underlying graph. Using this fact, we are able to obtain near-optimal lower bounds on the rank of SP refutations by combining Theorem 20 with Lemma 18 as in the proof of Theorem 19. This lower bound should be contrasted with the logarithmic-rank SP upper bound on Tseitin from Theorem 6.

▶ **Theorem 21.** *There a constant-degree graph G on n vertices such that if ℓ is any $\{0,1\}$ vertex labelling with odd total weight, the CNF formula* $\mathrm{Tseitin}(G,\ell)\circ\mathsf{VER}^n$, *on $O(n)$ variables and clauses, requires* SP *refutations of rank $\Omega(n/\log^2 n)$.*

5.3 SP Refutations Cannot Be Balanced

Optimistically, one could hope that the length and rank of SP refutations may be closely related and therefore that we could leverage these rank bounds to obtain lower bounds on the length of SP refutations. We answer this question negatively, showing that there exists a contradictory CNF which admits short refutations, but for which these refutations must be almost maximally deep. That is, we show that SP refutations cannot be balanced; an SP refutation of length S does not imply one of rank $O(\log S)$. This shows that in SP the rank of refutations is a distinct complexity measure from the length.

In order to obtain time-space tradeoffs, de Rezende et al. [11] proved Resolution upper bounds on the lifted pebbling contradictions. Combining this upper bound (which can be simulated efficiently in SP) with the lower bound from Theorem 19 exhibits a formula that requires small size, but near-maximal rank to refute in SP.

▶ **Theorem 22.** SP *refutations cannot be balanced.*

Proof. Suppose that a SP refutation of length S implied the existence of a SP refutation of the same formula of rank $O(\log S)$. Let G be the graph from Theorem 19 on n vertices, and let Peb_G be the pebbling contradictions defined on this graph. It follows immediately from Lemmas 7.2 and 7.3 from de Rezende et al. [11] that for any graph of indegree 2 on n vertices, that there is a Resolution refutation of $\mathrm{Peb}_G \circ \mathsf{IND}^n_\ell$ of length $O(n\ell^3)$. Since SP can p-simulate Resolution, this implies a $\mathsf{poly}(n)$ upper bound on the same formula in SP. Under the assumption that SP refutation can be balanced, this would imply a SP refutation of depth $O(\log n)$ of $\mathrm{Peb}_G \circ \mathsf{IND}^n_\ell$, contradicting the lower bound from Corollary 19. ◀

Although it is a well-known fact that tree-like Cutting Planes refutations cannot be balanced, Impagliazzo et al. [20] show that the randomized communication protocols for the search problem obtained from CP refutations can be balanced. Using this fact, they show that a length S tree-like Cutting Planes proof implies a depth $O(\log S)$ protocol for the search problem. This implies that communication cost lower bounds for the search problem imply length lower bounds for tree-like Cutting Planes refutations.

Optimistically one could hope that a similar approach could be applied to SP refutations. This is reinforced by the fact that the real communication protocols for the search problem obtained from SP refutations (Lemma 17) maintains the same topology as the refutation. That is, the cost and size of the resulting protocol are approximately equal to the rank and length of the proof (unfortunately, this is not the case for the randomized protocols obtained from SP refutations). Therefore, one might hope that even though SP cannot be balanced, the corresponding real communication protocols for the search problem can be balanced. Thus, lower bounds on the rank of real-communication protocols for the search problem would imply lower bounds on the size of SP proofs.

▶ **Corollary 23.** *Any* SP *refutation of length S and rank r of an unsatisfiable formula $\mathcal{F}$ implies a real communication protocol of size $O(S \cdot n)$ and cost $O(r + \log n)$ for* $\mathsf{Search}_{X,Y}(\mathcal{F})$.

This follows from observing that the protocol obtained in Lemma 17 also preserves the topology (and therefore both the rank and the length) of the refutation.

5.4 Real Communication Protocols Cannot Be Balanced

Analogous to Theorem 22 (SP proofs cannot be balanced) in this section we will show that real communication protocols cannot be balanced. This should be contrasted with other standard models of communication such as randomized and deterministic, which can be balanced. In particular, we exhibit a function which has a real communication protocol of small size, but for which every real protocol must have high cost. Towards this end, we prove lower bounds on the real communication complexity of the famous NP^{cc}-complete problem set disjointness function[1]

The set disjointness function DISJ_n defined as follows: each player is given an n-bit string, interpreted as indicator vectors for an underlying set of n elements, and they are asked to determine whether their sets are disjoint. That is, the players aim to solve the function $\mathrm{DISJ}_n(x, y) = \bigvee_{i\in[n]}(x_i \wedge y_i)$. To our knowledge, the only known technique for obtaining lower bounds on the real communication of any problem are via lifting theorems, reducing the task of proving lower bounds on lifted functions to the decision tree complexity of the un-lifted function. Although DISJ_n can be seen as a lifted function (the OR_n function lifted by the two-bit AND gadget), these real communication lifting theorems work only for super-constant sized gadgets, and therefore cannot be applied directly to DISJ_n. We circumvent this difficulty by exploiting the fact that DISJ_n is NP^{cc}-complete. To do so, we find a lifted function in NP^{cc} to which our simulation theorems can be applied. Consider the n-bit OR_n function composed with the index gadget, $\mathrm{OR} \circ \mathsf{IND}_\ell^n$, for some ℓ defined later.

▶ **Lemma 24.** $\mathrm{OR}_n \circ \mathsf{IND}_\ell^n \in \mathsf{NP}^{cc}$*, for any* $\ell \leq 2^{polylog(n)}$.

Proof. First, observe that the index gadget $\mathsf{IND}_\ell(x_i, y_i)$, for a single bit i of the input to the OR_n function, can be computed by a brute force protocol in $\log \ell$ bits of communication. Alice simply sends to Bob the $\log \ell$ bits of her input $x_i = x_{i,1}, \ldots, x_{i,\log \ell}$. Bob is then able to evaluate $\mathsf{IND}_\ell(x_i, y_i)$.

Now, consider the following NP^{cc} protocol for $\mathrm{OR}_n \circ \mathsf{IND}_\ell^n$: Alice and Bob are given as a proof, a $\log n$-bit string indicating the index $i \in [n]$ of the OR_n function where $\mathsf{IND}_\ell(x_i, y_i) = 1$. They then perform the brute force protocol to evaluate $\mathsf{IND}_\ell(x_i, y_i)$ and verify that the outcome is indeed 1. In total, this requires $\log \ell + \log n + 1 = polylog(n)$ bits of NP^{cc}-communication. ◀

To obtain lower bounds on $\mathrm{OR}_n \circ \mathsf{IND}_\ell^n$, we appeal to the real communication simulation theorem (Theorem 18), reducing communication lower bounds for $\mathrm{OR}_n \circ \mathsf{IND}_\ell^n$ on the lifted problem to the well-known linear decision tree lower bounds on the OR_n function.

▶ **Lemma 25.** *Let $\ell = n^4$. The cost of any real communication protocol computing* $\mathrm{OR}_n \circ \mathsf{IND}_\ell^n$ *is* $\Omega(n \log \ell)$.

Proof. Combining the $\Omega(n)$ lower bound on the decision tree complexity of computing the OR_n function with the simulation theorem of de Rezende et al. [11] proves the result. ◀

[1] Recall that the NP^{cc} communication complexity of a function f is the minimum size of any monochromatic rectangle cover of the communication matrix of f.

▶ **Theorem 26.** *The cost of any real communication protocol for* DISJ_n *is* $\Omega((n \log n)^{1/5})$.

Proof. Let $\ell = n^4$. Consider the following reduction from $\mathrm{OR} \circ \mathsf{IND}_\ell^n$ to an instance of set disjointness. By Lemma 24, the NP^{cc}-complexity of $\mathrm{OR}_n \circ \mathsf{IND}_\ell^n$ is $\log \ell + \log n + 1$. That is, there exists a cover of the 1s of the communication matrix of $\mathrm{OR}_n \circ \mathsf{IND}_\ell^n$ with at most $2n\ell$ rectangles. Enumerating the 1-rectangles gives us an instance of set disjointness: on input (x, y) to $\mathrm{OR}_n \circ \mathsf{IND}_\ell^n$, Alice constructs the $2n\ell$-bit string which is the indicator vector $I_x(x)$ of the set of 1 rectangles which x belongs to, similarly Bob constructs $I_y(y)$ the same for y. Thus $\mathrm{OR}_n \circ \mathsf{IND}_\ell^n(x, y) = 1$ iff $\mathrm{DISJ}_{2n\ell}(I_x(x), I_y(y)) = 1$. Combining this with the lower bound from Lemma 25 gives a lower bound of $\Omega((n \log \ell)^{1/5})$ on the cost of any real communication protocol computing DISJ_n. ◀

▶ **Corollary 27.** *Real communication protocols cannot be balanced.*

Proof. Observe that $\mathrm{DISJ}_n = \vee_{i=1}^n (x_i \wedge y_i)$ can be computed in a $2n + 1$-node protocol by Alice and Bob comparing x_i and $2 - y_i$ for $i = 1, \ldots, n$. Suppose by contradiction that one could balance real communication protocols. The size $2n + 1$ protocol would therefore imply a cost $\log(2n + 1)$ real protocol for DISJ_n, contradicting the lower bound from Theorem 26. ◀

6 Conclusions

This paper introduces and develops the Stabbing Planes proof system as a natural extension of DPLL and pseudoBoolean solvers to handle a more expressive set of queries. Although it is equivalent to a tree-like version of a system already in the literature, this new perspective turns out to be quite useful for proving upper bounds. This paper is only a preliminary exploration of the SP proof system and leaves open many interesting problems from both a theoretical as well as a practical perspective.

As mentioned in the preliminaries, we do not have an analog to Cook et al. [8] for Stabbing Planes and so it is unknown whether for SP refutations, the length and size (number of bits) can be treated as the same measure. That is, is it possible to prove that any SP refutation of *length* l can be simulated by an SP planes refutation of size $\mathsf{poly}(l, n)$?

We have shown that that CP refutations of small rank can be simulated by SP refutation of small rank, and that CP refutations of small size can also be simulated by SP refutations of small size. Can we simulate both rank and size efficiently? That is, can any CP refutation of rank r and size s be simulated by a SP refutation of rank $\mathsf{poly}(r)$ and size $\mathsf{poly}(s)$?

Is it possible to prove superpolynomial lower bounds for SP? Krajíček [23] gave exponential lower bounds on the length of R(CP) refutations when both the width of the clauses, and the size of the coeficients appearing in the inequalities are sufficiently bounded. This was later improved by Kojevnikov [22] to remove the restriction on the size of the coefficients for tree-like R(CP). In particular, to obtain any lower bound at all, the width of the clauses appearing in the R(CP) refutations must be bounded by $o(n/\log n)$. From Theorem 9, a size S and rank D SP refutation implies an R(CP) proof of size $O(S)$ and width $O(D)$. Therefore, this result is also a size lower bound for bounded-depth SP. Unfortunately, it appears that these techniques are fundamentally limited to be applicable only to SP refutations with low depth, and so new techniques seem needed to overcome this barrier.

As mentioned in the introduction, we feel that SP has potential, in combination with state-of-the-art algorithms for SAT and ILP, for improved performance on some hard instances, and problems such as maxSAT and counting satisfying assignments. The upper bound on the Tseitin example illustrates the kind of reasoning that SP is capable of: arbitrarily splitting the solution space into sub-problems based on some measure of progress. It would be interesting to build a SP-based solver, or to add SP-like branching to a solver such as CPLEX.

It has been a long-standing conjecture that CP does not have short refutations of the Tseitin formulas, as CP is unable to count mod2. On the other hand, Theorem 17 gives a quasi-polynomial upper bound on the Tseitin formulas in SP. Therefore, a natural approach to separating SP and CP is through proving CP lower bounds for Tseitin formulas.

References

1 Paul Beame, Noah Fleming, Russell Impagliazzo, Antonina Kolokolova, Denis Pankratov, Toniann Pitassi, and Robert Robere. Stabbing planes. *CoRR*, abs/1710.03219, 2017. arXiv:1710.03219.

2 Paul Beame, Trinh Huynh, and Toniann Pitassi. Hardness amplification in proof complexity. In *STOC'10*, pages 87–96, 2010.

3 Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004.

4 Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001.

5 Daniel Le Berre. Handling Pseudo-Boolean constraints in a CDCL solver: a practical survey. In *Dagstuhl Seminar 15171: Theory and Practice of SAT Solving*, April 2015.

6 Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM J. Comput.*, 30(5):1462–1484, 2000.

7 Joshua Buresh-Oppenheim, Nicola Galesi, Shlomo Hoory, Avner Magen, and Toniann Pitassi. Rank bounds and integrality gaps for cutting planes procedures. *Theory of Computing*, 2(4):65–90, 2006.

8 William Cook, Collette Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, 1987.

9 Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, jul 1962.

10 Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

11 Susanna F. de Rezende, Jakob Nordstrom, and Marc Vinyals. How limited interaction hinders real communication (and what it means for proof and circuit complexity). In *FOCS'16*, pages 295–304, 2016.

12 Yuval Filmus, Pavel Hrubes, and Massimo Lauria. Semantic versus syntactic cutting planes. In *STACS'16*, pages 35:1–35:13, 2016.

13 Noah Fleming, Denis Pankratov, Toniann Pitassi, and Robert Robere. Random cnfs are hard for cutting planes. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:45, 2017.

14 Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. In *STOC'14*, pages 847–856, 2014.

15 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *FOCS'15*, pages 1077–1088, 2015.

16 Pavel Hrubes and Pavel Pudlák. A note on monotone real circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:48, 2017.

17 Pavel Hrubeš and Pavel Pudlák. Random formulas, monotone circuits, and interpolation. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:42, 2017.

18 Trinh Huynh and Jakob Nordstrom. On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space trade-offs in proof complexity. In *STOC '12*, pages 233–248, 2012.

19 IBM ILOG. The CPLEX optimizer. URL: `https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`.

20 Russell Impagliazzo, Toniann Pitassi, and Alasdair Urquhart. Upper and lower bounds for tree-like cutting planes proofs. In *LICS '94*, pages 220–228, 1994.

21 Stasys Jukna. *Boolean function complexity : advances and frontiers.* Algorithms and combinatorics. Springer, 2012.

22 Arist Kojevnikov. Improved lower bounds for tree-like resolution over linear inequalities. In *SAT'07*, pages 70–79, 2007.

23 Jan Krajícek. Discretely Ordered Modules as a First-Order Extension of the Cutting Planes Proof System. *The Journal of Symbolic Logic*, 63(4):1582–1596, 1998.

24 Jan Krajícek. Interpolation by a Game. *Mathematical Logic Quarterly*, 44:450–458, 1998.

25 Saburo Muroga. *Threshold logic and its applications.* Wiley-Interscience, 1972.

26 Noam Nisan. The communication complexity of threshold gates. In *In Proceedings of "Combinatorics, Paul Erdos is Eighty*, pages 301–315, 1994.

27 Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62(3):981–998, 1997.

28 Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999.

29 Olivier Roussel and Vasco M Manquinho. Pseudo-Boolean and Cardinality Constraints. In *Handbook of satisfiability*, pages 695–733. IOS Press, 2009.

30 Günter M. Ziegler. *Lectures on Polytopes.* Springer-Verlag, New York, 1995.