Stabbing Planes

Paul Beame[†]
University of Washington

Noah Fleming[‡]
Memorial University

Russell Impagliazzo
UCSD

Antonina Kolokolova[‡]
Memorial University

Denis Pankratov[‡]
Concordia University

Toniann Pitassi[§]
Columbia University & IAS

Robert Robere[‡]
McGill University

March 17, 2023

Abstract

We develop a new semi-algebraic proof system called Stabbing Planes which formalizes modern branch-and-cut algorithms for integer programming and is in the style of DPLL-based modern SAT solvers. As with DPLL there is only a single rule: the current polytope can be subdivided by branching on an inequality and its "integer negation." That is, we can (non-deterministically 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 that Stabbing Planes can efficiently simulate the Cutting Planes proof system, and is equivalent to a tree-like variant of the $\mathsf{R}(\mathsf{CP})$ system of Krajíček [54]. As well, we show that it possesses short proofs of the canonical family of systems of $\mathbb{F}_2$-linear equations known as the Tseitin formulas. Finally, we prove linear lower bounds on the rank of Stabbing Planes refutations by adapting lower bounds in communication complexity and use these bounds in order to show that Stabbing Planes proofs cannot be balanced. In doing so, we show that real communication protocols cannot be balanced and establish the first lower bound on the real communication complexity of set disjointness.

[†]Research supported by NSF Grants No. CCF-152424 and CCF-2006359.
[‡]Research supported by NSERC.
[§]Research supported by NSERC, NSF Grant No. CCF-1900460 and the IAS school of mathematics.
A preliminary version of this work appeared at the 9th *Innovations in Theoretical Computer Science* (ITCS).

1 Introduction

Proof complexity provides an effective and principled way to analyze classes of practical algorithms for solving NP-hard problems. The general idea is to formalize a class of algorithms as a proof system — a set of sound rules for making logical inferences — by extracting out the types of reasoning used in the algorithms. This allows us to discard the practical implementation details while still maintaining the techniques that the algorithm can employ. Thus, by proving lower bounds on the size of proofs in these proof systems, we obtain lower bounds on the runtime of the associated class of algorithms.

An illustrative example is the *DPLL* algorithm [29, 30], which forms the basis of modern conflict-driven clause learning algorithms for solving SAT. For a CNF formula F, the DPLL algorithm is the following recursive search algorithm for a satisfying assignment: choose a variable x_i (non-deterministically, or via some heuristic), and then recurse on the formulas $F \restriction (x_i = 0)$ and $F \restriction (x_i = 1)$. If at any point a satisfying assignment is found, the algorithm halts and outputs the assignment. Otherwise, if the current partial assignment falsifies some clause C of F, the recursive branch is terminated. If every recursive branch terminated with a falsified clause, then F is unsatisfiable and we can take the recursion tree as a proof of this fact; in fact, such a DPLL tree is equivalent to a *treelike resolution* refutation of F.

Beyond DPLL, the approach of using proof complexity for algorithm analysis has been successfully employed to study many other classes of algorithms. This includes *Conflict-driven clause-learning* algorithms for SAT [52, 61, 72], which can be formalized using *resolution* proofs [30]; lift-and-project methods for solving integer programming problems, which are formalized by the *Lovász-Schrijver* [60], *Sherali-Adams* [71], and *Sums-of-Squares proofs* [7, 45] proof systems; and the classical *cutting planes* methor for integer programming [18, 41], which is formalized by *Cutting Planes proofs* [18, 19, 23].

In this work, we continue the study of algorithms for solving integer programming problems through the lens of proof complexity. Many classic NP-optimization problems are naturally phrased using integer programming and, due to this, algorithms for integer programming have had a profound effect throughout computer science and beyond. Recall that in an integer programming problem, we are given a polytope $P \subseteq \mathbb{R}^n$ and a vector $c \in \mathbb{R}^n$, and our goal is to find a point $x \in P \cap \mathbb{Z}^n$ maximizing $c \cdot x$. A classic approach for solving integer programming problems is to refine the polytope P by introducing Chvátal-Gomory (CG) *cutting planes* [41]. A CG cutting plane for P is any inequality $ax \leq \lfloor b \rfloor$, where a is an integral vector, b is a rational vector, and every point in P satisfies $ax \leq b$. Observe that a CG cutting plane removes non-integral points from P while preserving integer points. Thus, with each additional cutting plane, the polytope becomes a better approximation to the integer hull.

Chvátal observed that the cutting planes method can be naturally formalized as a proof system. The *Cutting Planes* proof system proves the integer-infeasibility of a polytope P by deducing the empty polytope by a sequence of CG-cutting planes. This has led to a number of strong lower bounds on the runtime of these algorithms [13,35,48,67], as well as bounds on related measures such as the Chvátal rank (see for example [14,17,43]). However, while Cutting Planes has become a highly influential proof system in proof complexity, the original cutting planes algorithms suffer from numerical instabilities and difficulties choosing good heuristics, and are therefore seldom used on their own in practice.

Modern algorithms for integer programming combine cutting planes with a branch-and-bound procedure [5,57], resulting in a class of optimization algorithms known as *branch-and-cut algorithms* [65]. These algorithms search for an integer solution to a polytope P by the following two procedures

- *Branch.* Split P into smaller polytopes $P_1, \ldots, P_k$ such that every integer solution to P lies in at least one of $P_1, \ldots, P_k$.
- *Cut.* Refine $P_1, \ldots, P_k$ by introducing additional cutting planes.

Finally, the algorithm recurses on each of the resulting polytopes. While this branching rule is extremely general, in practice branching is typically done by single variables: selecting a variable x_i and branching on all possible integer outcomes $P \cap \{x_i = t\}$ for each feasible integer value t. Other schemes that have been used in practice include branching on the hamming weight of a subset of variables [34] or branching using basis-reduction techniques [1, 2,

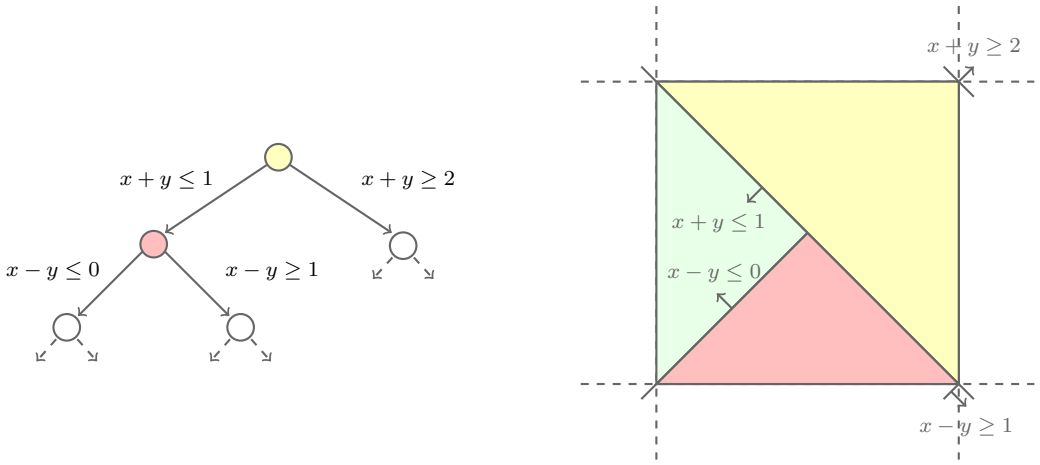

Figure 1: A partial stabbing planes proof (left) and its result on the unit square (right). The yellow and red areas are removed from the polytope (green), and we recurse on both sides.

46, 51, 56, 59][*]

While these branch-and-cut algorithms are much more efficient in practice than the classical cutting planes methods they are no longer naturally modelled by Cutting Planes proofs. In this work we introduce[†] the *Stabbing Planes* (SP) proof system in order to properly model branch-and-cut solvers. Intuitively, Stabbing Planes has the same branching structure as DPLL, but generalizes branching on single variables to branching on integer linear inequalities.

We formalize Stabbing Planes in stages as a generalization of DPLL. Recall that in the setting of integer programming we want to prove the integer-infeasibility of a system of integer linear inequalities $Ax \geq b$ over real-valued variables. Further, suppose for simplicity of exposition that the system encodes a CNF formula F. We can rephrase DPLL *geometrically* to the setting of integer linear programming. Consider some DPLL refutation of F. Each time the DPLL tree branches on the $\{0, 1\}$-value of a variable x_i, instead branch on whether $x_i \leq 0$ or $x_i \geq 1$; because the encoding $Ax \geq b$ of F includes axioms $x_i \geq 0$ and $x_i \leq 1$ this is equivalent. After this replacement, each node v in the DPLL tree is naturally associated with a polytope P_v of points satisfying $Ax \geq b$ and each of the inequalities labelling the root-to-v path. Since we began with a DPLL refutation, it is clear that for any leaf ℓ, the polytope P_ℓ associated with the leaf is empty, as any $\mathbb{Z}^n$-valued point would have survived each of the inequalities queried on some path in the tree and thus would exist in one of the polytopes at the leaves.

The Stabbing Planes system is then the natural generalization of the previous object: at each step in the refutation an *arbitrary* integer linear form ax in the input variables is chosen and we recurse assuming that it is at least some integer b or at most its *integer negation* $b - 1$. Observe that because a and b are integral, any $x^* \in \mathbb{Z}^n$ will satisfy at least one of the inequalities $(ax \leq b-1,\ ax \geq b)$, and so if the polytope at each leaf (again, obtained by intersecting the original system with the inequalities on the path to this leaf) is empty then we have certified that the original system has no integral solutions (cf. Figure 1). The queries in a Stabbing Planes proof correspond to what is known as branching on *general disjunctions* or *split disjunctions* [22] in integer programming, and capture the majority of branching that is done in practice [20]. One of the major advantages of Stabbing Planes is its simplicity: refutations are decision trees that query integer linear inequalities.

Recall that branch-and-cut solvers combine Stabbing Planes-style branching with additional cutting planes in order to refine the search space. We show that a single branching step in a Stabbing Planes proof can actually simulate CG cuts to the polytope, and therefore Stabbing Planes can simulate the execution of a branch-and-cut solver. A novel corollary is that Stabbing Planes can polynomially-simulate dag-like Cutting Planes despite being a tree-like

[*]For an in-depth discussions on branch-and-bound and branch-and-cut we refer the reader to [3, 20, 73].

[†]Recently (and subsequent to the conference publication of this work [10]), the authors became aware of the invited chapter of Pudlák [68] for the Logic Colloquium'97, in which he states that an unpublished work of Chvátal defines the Stabbing Planes system and the simulation of Cutting Planes. We take this independent discovery as evidence that Stabbing Planes is a highly natural proof system, which deserves further exploration.

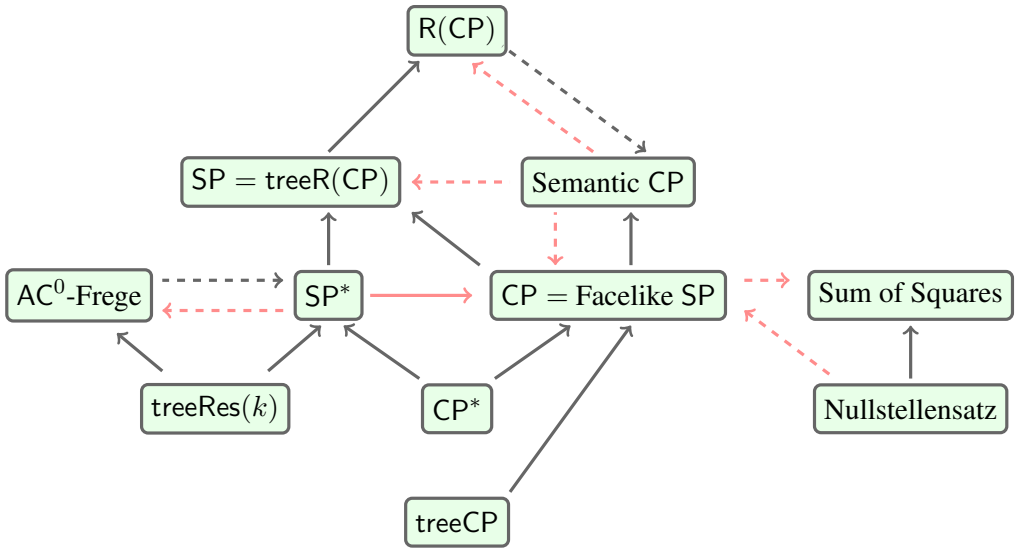

Figure 2: Known relationships between relevant proof systems. A solid black (red) arrow from proof system P_1 to P_2 indicates that P_2 can polynomially (quasi-polynomially) simulate P_1. A black (red) dashed arrow from P_1 to P_2 indicates that P_2 cannot polynomially (quasi-polynomially) simulate P_1.

refutation system. This simulation was extended by [8] to show that Stabbing Planes can simulate *disjunctive cuts* which capture the vast majority of cutting planes used in practice, including *lift and project cuts* [6], *split cuts* [22], *Gomory mixed integer* cuts [40] and *MIR cuts*.

Beyond providing a theoretical model for branch and cut algorithms, we believe that the simplicity of Stabbing Planes proofs, as well as its closeness to DPLL, makes Stabbing Planes a better starting point for the analysis and development of search algorithms based on integer programming, such as *pseudoboolean* SAT solvers, than established proof systems. From the perspective of SAT solving, even though treeRes is equivalent to DPLL, it is the search point of view of DPLL that has led to major advances in SAT algorithms. 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 deductive rules of resolution. Stabbing Planes offers similar benefits with respect to reasoning about inequalities. Furthermore, Stabbing Planes is a direct generalization of DPLL, and therefore we hope that the fine-tuned heuristics that have been developed for modern DPLL-based solvers can be lifted to algorithms based on Stabbing Planes.

Stabbing Planes in Proof Complexity

Despite its simplicity, Stabbing Planes proofs are remarkably powerful. As a motivating example, we give simple, short (quasi-polynomial size) SP proofs of systems of $\mathbb{F}_2$ linear equations known as the *Tseitin formulas*. These formulas are one of the canonical hard examples for many algebraic and semi-algebraic proof systems, including Nullstellensatz [44], Polynomial Calculus [15], and Sum-of-Squares [45, 70].

Theorem 1. *There are quasipolynomial size Stabbing Planes proofs of any instance of the Tseitin tautologies.*

We also explore the relationships between SP and other proof systems. This is summarized in Figure 2. Most notably, we show that SP is polynomially-equivalent to treeR(CP), the tree-like variant of Krajíček's R(CP) proof system [54]. This system can be thought of as a mutual generalization of resolution and Cutting Planes, in which the lines are disjunctions of integer linear inequalities, and we are allowed to apply Cutting Planes rules on the inequalities and resolution-style cuts on the disjunctions.

Theorem 2. *The proof systems* SP *and* treeR(CP) *are polynomially equivalent.*

We note that even though SP is equivalent to a system already in the literature, the new perspective provided by it is indeed enlightening as none of the results in this section were known for treeR(CP) (including the simulation of CP!).

The remainder of this work tackles the problem of proving superpolynomial lower bounds on the complexity of SP proofs. Although we are unable to establish size lower bounds, we prove nearly optimal lower bounds on the *depth* of SP refutations, as well as explain why several natural approaches for proving size bounds fail. The depth of an SP proof — the longest root to leaf path — is a natural parameter that captures the parallelizability of proofs, and is closely related to *rank* measures of polytopes, which have been heavily studied in integer programming theory [17].

Theorem 3. *There exists a family of unsatisfiable CNF formulas $\{F_n\}$ for which any* SP *refutation requires depth* $\Omega(n/\log^2 n)$

The proof of this theorem proceeds by showing that shallow SP proofs give rise to short randomized and real communication protocols for the *false clause search problem*, and then appealing to known lower bounds for this problem. In many strong proof systems such as Frege and Extended Frege it is known that depth lower bounds imply size lower bounds via so-called *balancing theorems* [55]. However, we can show that SP proofs cannot be balanced. More precisely, SP proofs of size s do not imply the existence of proofs of size $\mathsf{poly}(s)$ and depth $\mathsf{polylog}(s)$.

While SP proofs cannot be balanced, the real communication protocols that result from SP proofs preserve the topology of the SP proof, and therefore size lower bounds on SP would follow by showing instead that the real communication protocols themselves could be balanced. There is a precedent for this: both deterministic and randomized communication protocols *can* be balanced, and lower bounds on treeCP proofs were obtained by exploiting this fact [49]. However, we can also show that real communication protocols cannot be balanced. Enroute, we establish the first superlogarithmic lower bound on the real communication complexity of the set disjointness function, a function which has been central to many of the lower bounds which exploit communication complexity.

1.1 Related and Subsequent Work

Lower Bounds on Variable Branching. Lower bounds for a number of branch-and-cut algorithms using *variable-branching* — meaning that they branch on the integer value of single variables, rather than arbitrary inequalities (i.e., DPLL which branches on $x_i \in \mathbb{Z}$) — have previously been established. The first example of this was the lower bound of Jerslow [50] on the number of queries made by branch-and-bound algorithms with variable branching. Cook and Hartman proved exponential lower bounds on the number of operations required to solve certain travelling salesman instances by branch-and-cut algorithms which use variable branching and Chvátal-Gomory cuts [24]. However, their lower bound is exponential only in the number of variables, and not in the number of inequalities. The first truly exponential (in the encoding size of the instance) lower bound for branch-and-cut algorithms which use variable branching was established by Dash [27] by extending the lower bound of Pudlák [67] for Cutting Planes proofs.

Lower Bounds for treeR(CP). Lower bounds on certain restrictions of treeR(CP) were established in earlier works. Krajíček [54] proved superpolynomial lower bounds on the size of R(CP) proofs when both the *width* of the clauses and the magnitude of the coefficients of every line are sufficiently bounded. Concretely, letting w and c be upper bounds on the width and coefficient size respectively, the lower bound that he obtained was $2^{n^{\Omega(1)}}/c^{w\log^2 n}$. For treeR(CP), Kojevnikov [53] improved this lower bound to $\exp(\Omega(\sqrt{n/w\log n}))$, removing the dependence on the coefficient size. In Figure 5.1 we prove a size-preserving simulation of SP by treeR(CP) which translates depth d SP proofs into width d treeR(CP) proofs. Therefore, Kojevnikov's result implies a superpolynomial lower bound on the size of SP proofs of depth $o(n/\log n)$. In Subsection 6.1 we exhibit a formula for which any SP refutation requires depth $\Omega(n/\log^2 n)$.

Subsequent Work. Following the conference version of this work [10], Dadush and Tiwari [25] showed that the Stabbing Planes proofs of the Tseitin formulas could be efficiently translated into CP. This refuted a long-standing conjecture that CP requires exponential size refutations of these formulas [23]. In the same paper, they established a polynomial equivalence between the number of nodes and the *size* of SP proofs (i.e., the number of bits needed to express the proof).

Dadush and Tiwari also considered SP in the context of *mixed integer programming* (MIP) and proved exponential lower bounds on SP in this setting. In this setting, you are given a polytope $P = \{(x, y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} : Ax + By \geq b\}$, and you are searching for an integer solution to the x-variables and a real solution to the y-variables. In other words, rather than proving that $P \cap \mathbb{Z}^n = \emptyset$, instead you would like to prove that $P \cap \mathbb{Z}^{n_1} \cap \mathbb{R}^{n_2} = \emptyset$. In this case, SP queries involving y-variables are disallowed, as this would not be sound. As shown by Dadush and Tiwari, this restriction turns out to be enough to obtain quite simple proofs of intractability. First, they prove that any SP refutation of a certain system of 2^n many inequalities (encoding the *complete unsatisfiable formula*) requires size $2^n/n$. Next, they show that this system of inequalities admits a $\mathsf{poly}(n)$-size MIP *extended formulation*. As SP cannot branch on the extension variables, refuting this extended formulation is identical refuting the complete unsatisfiable formula in SP. Dey, Dubey, and Molinaro [32] extended this technique to prove lower bounds for a number of MIP instances for packing, set cover, the Travelling Salesman problem, and the cross polytope, even when Gaussian noise is added to the coefficients. However, this technique crucially relies on the fact that for MIP problems, SP queries cannot involve the real-variables, and therefore it does not appear to be possible to extend this technique to prove lower bounds on pure integer programming problems (i.e. those with only integer-variables) such as standard encodings of CNF formulas.

Fleming et al. [35] showed that any Stabbing Planes proof with quasipolynomially bounded coefficients (SP^*) can be translated into a Cutting Planes proof with at most a quasipolynomial increase in size. This allowed them to lift the exponential lower bounds on Cutting Planes proofs [36, 39, 48, 67] to SP^*, and even to SP proofs with coefficients of size $\exp(n^\delta)$ for some constant $\delta < 1$. As well, using this connection, they generalized the result of Dadush and Tiwari to show that there are quasipolynomial-size Cutting Planes refutations of *any* unsatisfiable system of linear equations over the finite field $\mathbb{F}_p$ for any prime p. To prove this simulation of SP^* by Cutting Planes, they characterized Cutting Planes as a subsystem of Stabbing Planes, which they called *facelike* Stabbing Planes. Briefly, a facelike Stabbing Planes proof restricts Stabbing Planes queries to have one side of the query be a face of the current polytope. Then, they show that SP proofs can be made facelike with a blowup that is proportional to the size of the coefficients and the diameter of the polytope.

Basu et al. [8] showed that SP can simulate *disjunctive cuts*, a result which we cover in more detail in Subsection 4.3. Furthermore, they give an IP instance can be solved in size $O(1)$ in SP but requires $\mathsf{poly}(n)$ deductions using split cuts. As well, they explore the effect that *sparsity* —the number of non-zero coordinates in each query — has on branch-and-cut. Sparse queries can be thought of as an intermediate between full SP branching and variable branching. They provide an instance where the any branch-and-bound tree must be of exponential size if the sparsity is $o(n)$.

Recently, Dantchev et al. [26] introduced several novel techniques for proving lower bounds on SP proofs by exploiting their geometric structure. In particular, they make use of the fact that for a polytope P, every point $x^* \in P$ must be contained within some slab of the SP proof. This allowed them to establish linear lower bounds on the size of SP proofs of the *pigeonhole principle* as well as the *Tseitin formulas*; because SP proofs are binary trees, this leads to a depth $\Omega(\log n)$ lower bound for both formulas.

1.2 Organization

We begin by formally defining the Stabbing Planes proof system in Section 2. The Stabbing Planes refutations of the Tseitin formulas are given in Section 3. Section 4 explores how Stabbing Planes relates to Cutting Planes: we show that Stabbing Planes can polynomially simulate Cutting Planes, and we explore whether a simulation of Cutting Planes by Stabbing Planes can preserve other parameters of the proof in Subsection 4.2. We end this section (in Subsection 4.3) by observing that Stabbing Planes can simulate most other types of cutting planes that are used in integer programming. In Section 5 we explore how Stabbing Planes compares to other popular proof systems in the literature, and we prove the equivalence with $\mathsf{treeR}(\mathsf{CP})$. In the final section (Section 6) we explore whether we can prove lower bounds on Stabbing Planes with unbounded coefficients. We prove unrestricted depth lower bounds in Subsection 6.1 and rule out several natural approaches that utilize communication complexity in Subsection 6.2.

2 Preliminaries

We begin by formally defining the Stabbing Planes proof system. For this, it will convenient to use the following combination of Farkas' Lemma with Carathéodory's Theorem, which we state next.

Farkas' Lemma. *Let $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$. Then exactly one of the following holds:*

(i) There exists $x \in \mathbb{R}^n$ such that $Ax \leq b$.

(ii) There exists $y \in \mathbb{Z}^m$ with $y \geq 0$ such that $y^\top A = 0$ and $y^\top b < 0$. Moreover, y has at most $n + 2$ non-zero coordinates.

The "moreover" part in (ii) follows from Carathéodory's Theorem. Proofs of both Carathéodory's Theorem and Farkas' Lemma can be found in [20].

With this, we are ready to define the Stabbing Planes proof system. The Stabbing Planes proof system is a proof system for refuting the existence of integer-solutions to systems of linear inequalities; we will call such systems *unsatisfiable*.

Stabbing Planes. Let $Ax \geq b$ be an unsatisfiable system of linear inequalities. A *Stabbing Planes (*SP*) refutation* of $Ax \geq b$ is a directed binary tree, T, where each edge is labelled with a linear integral inequality satisfying the following *consistency conditions*:

- *Internal Nodes.* For any internal node u of T, if the right outgoing edge of u is labelled with $cx \geq d$, then the left outgoing edge is labelled with its *integer negation* $cx \leq d - 1$.
- *Leaves.* Each leaf node v of T is labelled with a conic combination of inequalities in F with inequalities along the path leading to v that yields $0 \geq 1$ (provided by Farkas' Lemma).

For an internal node u of T, the pair of inequalities $(cx \leq d - 1,\ cx \geq d)$ is called the *query* corresponding to the node. Every node of T has a polytope P associated with it, where P is the polytope defined by the intersection of the inequalities in F together with the inequalities labelling the path from the root to this node. We will say that the polytope P *corresponds* to this node. That is, if P is the polytope corresponding to a node v which queries $(cx \leq d - 1,\ cx \geq d)$, then the polytopes of the children of v are given by $P \cap \{x \in \mathbb{R}^n : cx \leq d - 1\}$ and $P \cap \{x \in \mathbb{R}^n : cx \geq d\}$. For readability, we will use the abbreviation $P \cap \{cx \geq d\}$ for $P \cap \{x \in \mathbb{R}^n : cx \geq d\}$.

The *slab* corresponding to the query is $\{x^* \in \mathbb{R}^n \mid d - 1 < cx^* < d\}$, which is the set of points ruled out by this query. The *width* of the slab is the minimum distance between $cx \leq d - 1$ and $cx \geq d$, which is $1/\|c\|_2$. This gives an intuitive geometric interpretation of SP refutations: at each step we remove a slab from the polytope and recurse on the resulting polytopes on both sides of the slab. The aim is to recursively cover the polytope with slabs until every feasible point has been removed. An example of this can be seen in Figure 1, where the yellow and red areas are the slabs of the two queries.

The *size* of an SP refutation is the bit-length needed to encode a description of the entire proof tree, which, for CNF formulas as well as sufficiently bounded systems of inequalities, is polynomially equivalent to the number of queries in the refutation. In particular, Dadush and Tiwari [25] prove the following.

Proposition 4 (Corollary 1.2 in [25])**.** *Let $Ax \geq b$ be any unsatisfiable system of linear equations whose coefficients require ℓ bits to express, and let s be the number of nodes in an* SP *refutation of $Ax \geq b$. Then there exists an* SP *refutation of size $s\ell n^6$.*

As well, the *depth* (or *rank*) of the refutation is the depth of the binary tree. The depth of refuting an unsatisfiable system of linear inequalities $Ax \geq b$, denoted $\mathsf{depth}_{\mathsf{SP}}(Ax \geq b)$, is the minimum depth of any SP refutation of $Ax \geq b$. Observe that any unsatisfiable system of inequalities $Ax \geq b$ whose corresponding polytope is contained within the unit cube $[0, 1]^n$ (this includes the encodings of all CNF formulas) has a trivial size 2^n and depth n SP refutation by branching on $(x_i \leq 0,\ x_i \geq 1)$ for every $i \in [n]$.

We will be particularly interested in Stabbing Planes refutations of unsatisfiable CNF formulas. Given a CNF formula F we can translate it into an equisatisfiable system of linear inequalities in the natural way. First, introduce the inequalities $0 \leq x_i \leq 1$ for every variable x_i. Second, for each clause $\bigvee_{i \in I} x_i \vee \bigvee_{j \in J} \neg x_j$ introduce the inequality

$$\sum_{i \in I} x_i + \sum_{j \in J} (1 - x_j) \geq 1.$$

It is easy to see that the this system of inequalities will have no integer solutions if and only if the original formula F was unsatisfiable. With this translation we consider Stabbing Planes refutations of CNF formulas F to be refutations of the translation of F into a system of linear inequalities.

With the previous translation in hand we show that SP is indeed a propositional proof system as defined by Cook and Reckhow [21].

Proposition 5. *Stabbing Planes is sound, complete, and polynomially verifiable.*

Proof. Completeness follows immediately from the fact that SP simulates DPLL, which is itself a complete proof system. Soundness follows because each slab in an SP proof, corresponding to a query $(cx \leq d-1,\ cx \geq d)$, removes only non-integral points. Indeed, by the integrality of c and d, any $x \in \mathbb{Z}^n$ satisfies either $cx \leq d-1$ or $cx \geq d$. Finally, to see that SP proofs are polynomially verifiable, observe that we only need to verify that every query is of the form $(cx \leq d-1,\ cx \geq d)$ for integral c and d, and that each conic combination labelling the leaves evaluates to $0 \geq 1$. □

We will also be interested in the *Cutting Planes* proof system, the first proof system to formalize a class of integer programming algorithms.

Cutting Planes. A *Cutting Planes proof* (CP) of an inequality $cx \geq d$ from a system of integer linear inequalities $Ax \geq b$ is a sequence of inequalities $\{c_i x \geq d_i\}_{i \in [s]}$ such that $c_s = c$, $d_s = d$, and each inequality $c_i x \geq d_i$ either belongs to $Ax \geq b$ or is deduced from earlier inequalities in the sequence by one of the following inference rules

- *Linear Combination.* From inequalities $c_i x \geq d_i, c_j x \geq d_j$, deduce any non-negative linear combination with integer coefficients.
- *Division.* From an inequality $c_i x \geq d_i$, if $t \in \mathbb{Z}$ divides all entries in c_i then deduce $(c_i/t)x \geq \lceil d_i/t \rceil$.

The *size* of a Cutting Planes derivation is the number of inequalities s, and is known to be equivalent up to a polynomial blow-up to the complexity of expressing the proof [23]. It is useful to visualize the derivation as a directed acyclic graph, where the nodes are the inequalities in the derivation, and for each inference, there are arcs from the at-most-two inequalities from which it was derived. With this in mind, the *depth* of a Cutting Planes derivation is the length of the longest root-to-leaf path in the dag. Finally, a Cutting Planes *refutation* of a system of integer linear inequalities $Ax \geq b$ is a derivation of the trivially false inequality $-1 \geq 0$.

3 Refutations of the Tseitin Formulas

As a motivating example, we show that Stabbing Planes has quasipolynomial size proofs of the *Tseitin formulas*, proving Theorem 1. For any graph $G = (V, E)$ and any labelling $\ell : V \to \{0, 1\}$ of the vertices, the *Tseitin formula* of (G, ℓ) is the following system of $\mathbb{F}_2$-linear equations: for each edge e we introduce a variable x_e, and for each vertex v we have an equation

$$\bigoplus_{u : uv \in E} x_{uv} = \ell(v),$$

asserting that the sum of edge variables incident to v must agree with its label $\ell(v)$. It is not difficult to see that a Tseitin formula is unsatisfiable iff $\sum_{v \in V} \ell(v)$ is odd. If we denote by $\deg(G)$ the maximum degree of any vertex in G then the Tseitin formula of (G, ℓ) can be encoded as a CNF formula with $|V| \cdot 2^{\deg(G)-1}$ many clauses. The next theorem shows that there is a quasi-polynomial size Stabbing Planes refutation of any Tseitin formula.

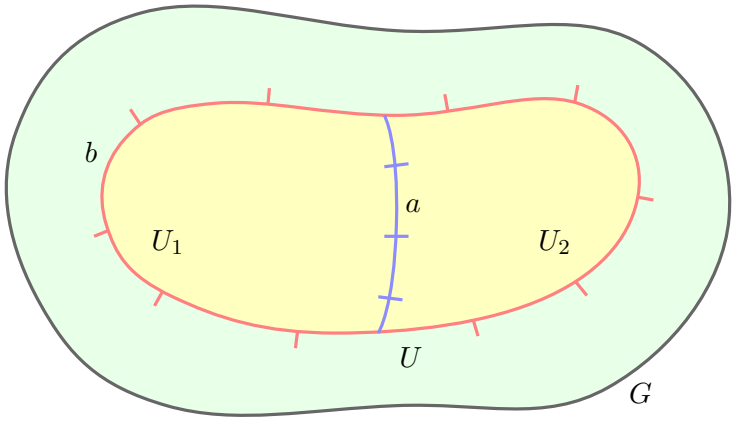

Figure 3: A single round of the algorithm. Note that $\kappa_{U_1} = a + b$ and $K_{U_2} = a + (K_U - b)$.

Theorem 6. *For any Tseitin instance (G, ℓ) there is a Stabbing Planes refutation of size $2^{O(\log^2 n)+\deg(G)}$ and depth $\deg(G) \cdot \log^2 n$.*

First we will introduce some notation. For $U \subseteq V$ let $\ell(U) = \oplus_{v \in U} \ell(v)$. As well, if $U, W \subseteq V$ then let $E[U, W]$ denote the set of edges with one endpoint in U and the other in W, and let $E[U]$ denote the set of edges with one endpoint in U.

Proof. The Stabbing Planes proof will implement the following recursive search algorithm for a violated constraint of the Tseitin instance. In each round we maintain a subset $U \subseteq V$ and an integer $\kappa_U \in \mathbb{N}$ representing the total value of the edges $E[U]$ leaving U. Over the algorithm, we maintain the invariant that $\ell(U) + \kappa_U$ is odd, which implies that there is a contradiction to the Tseitin instance inside of U.

Initially, set $U := V$ and $\kappa_V = 0$, and note that the invariant holds since $\ell(V)$ is odd by definition. Then perform the following algorithm (see also Figure 3):

1. Choose a balanced partition $U = U_1 \cup U_2$ (so $||U_1| - |U_2|| \leq 1$)
2. Query the value of $a = \sum_{e \in E[U_1, U_2]} x_e$ and $b = \sum_{e \in E[U_1] \setminus E[U_1, U_2]} x_e$.
3. The value of the edges leaving U_1 is $\kappa_{U_1} := a + b$ and the value of the edges leaving U_2 is $\kappa_{U_2} := a + (\kappa_U - b)$; so we recurse on the subset that maintains the invariant.

First we note that exactly one of the two subsets must maintain the invariant. This follows from the next short calculation:

$$\begin{aligned} \ell(U) + \kappa_U &= \ell(U_1) + \ell(U_2) + \kappa_U + 2a && \pmod 2 \\ &= (\ell(U_1) + a + b) + (\ell(U_2) + a + (\kappa_U - b)) && \pmod 2 \\ &= (\ell(U_1) + \kappa_{U_1}) + (\ell(U_2) + \kappa_{U_2}) && \pmod 2 \end{aligned}$$

thus, since $\ell(U) + \kappa_U$ is odd, it follows that exactly one of the $\ell(U_1) + \kappa_{U_1}$ or $\ell(U_2) + \kappa_{U_2}$ must also be odd. Second, we note that the recursion ends when $|U| = 1$, at which point we obtain an immediate contradiction between κ_U and the equation corresponding to the single node inside U.

To implement this algorithm in Stabbing Planes, it suffices to show how to perform the queries in step 2, and how to deduce a contradiction when $|U| = 1$; we begin with step 2. The first query can be performed by a binary tree with $|E[U_1, U_2| \leq n$ leaves, one corresponding to each possible query outcome. Internally, the tree queries all possible integer values for the sum (e.g. $(a \leq 0,\ a \geq 1), (a \leq 1,\ a \geq 2), \ldots$). The second query can similarly be performed by a tree with $|E[U_1]| \leq n$ leaves. Since we choose a balanced partition in each step, the recursion terminates in at most $O(\log n)$ rounds — thus, we have a tree with branching factor $O(n)$ and depth $O(\log n)$, yielding a size bound

of $n^{O(\log n)}$. Furthermore, each query can be implemented in a tree of depth $O(\log n)$, and so the depth of the proof is $O(\log^2 n)$.

For the leaf-case, when $|U| = 1$, let u be the unique vertex in U. Stabbing Planes has deduced that

$$\sum_{v:uv\in E} x_{uv} = \kappa_U \neq \ell(u) \pmod 2.$$

This is a contradiction to the Tseitin axiom $\oplus_{v:uv\in E} x_{uv} = \ell(v)$. However, the Tseitin formula is presented to SP as a system of linear inequalities (encoding a CNF formula) which is equivalent to the Tseitin formulas (a system of $\mathbb{F}_2$ linear equations) over integer solutions. Therefore, while there are no integer solutions satisfying $\kappa_U = \sum_{v:uv\in E} x_{uv} = \ell(u)$, there could still be *non-integer* solutions. To handle this, we simply force each of the $\deg(G)$ variables involved in this constraint to take integer values by sequentially querying the value of each variable one-by-one. That is, for each $\{x_{uv} : uv \in E\}$ we query $(x_{uv} \leq 0,\ x_{uv} \geq 1)$, noting that we have axioms saying that $x_e \geq 0$ and $x_e \leq 1$ for every $e \in E$. This can be done in a binary tree of height $\deg(G)$ with at most $2^{\deg(G)}$ leaves, where at each leaf we derive $0 \geq 1$. □

Together with the lower bounds of Buresh-Oppenheim et. al. [14] and Fleming et. al. [35] on the depth of Cutting Planes and semantic Cutting Planes proofs of the Tseitin formulas, Theorem 1 provides an exponential separation in terms of *depth* for these proof systems and Stabbing Planes.

4 The Relationship Between Stabbing Planes and Cutting Planes

It is an interesting question how Stabbing Planes compares to Cutting Planes, the main proof system based on ideas from integer programming. By contrasting the two systems we see three major differences:

- *Top-down vs. Bottom-up.* Stabbing Planes is a *top-down* proof system formed by performing queries on the polytope and recursing; while Cutting Planes is a *bottom-up* proof system, formed by deducing new inequalities from previously deduced ones.
- *Polytopes vs. Halfspaces.* Individual "lines" in a Stabbing Planes proof are polytopes, while individual "lines" in a Cutting Planes proof are halfspaces.
- *Tree-like vs. dag-like.* The graphs underlying Stabbing Planes proofs are trees, while the graphs underlying Cutting Planes proofs are general dags: intuitively, this means that Cutting Planes proofs can "re-use" their intermediate steps, while Stabbing Planes proofs cannot.

When taken together, these facts suggest that Stabbing Planes and Cutting Planes could be incomparable in power, as polytopes are more expressive than halfspaces, while dag-like proofs offer the power of line-reuse. Going against this intuition, we show next that Stabbing Planes and Cutting Planes are in fact very closely related.

4.1 Stabbing Planes Simulates Cutting Planes

A Cutting Planes proof is of a linear inequality $cx \geq d$ from polytope P, presented as a list of integer linear inequalities $\{a_i x \geq b_i\}$, is a sequence of inequalities $\{c_i x \geq d_i\}_{i\in[s]}$ such that the final inequality is $cx \geq d$, and each $c_i x \geq d_i$ is either one of the inequalities of P, or is deduced from previously derived inequalities by one of the following two deduction rules:

- *Conic Combination*. From inequalities $ax \geq b$, $cx \geq d$ deduce any nonnegative linear combination of these two inequalities with integer coefficients.
- *Division*. From an inequality $ax \geq b$, if $d \in \mathbb{Z}$ with $d \geq 0$ divides all entries of a then deduce $(a/d)x \geq \lceil b/d \rceil$.

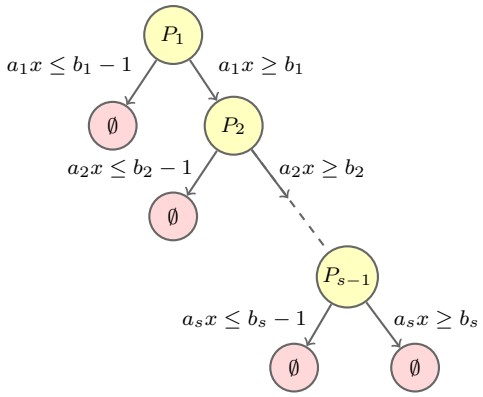

Figure 4: The Stabbing Planes refutation which results from translating a Cutting Planes refutation $P = P_1, \ldots, P_s = \emptyset$.

A Cutting Planes *refutation* is a proof of the trivially false inequality $0 \geq 1$.

Equivalently, we can view a Cutting Planes refutation as a sequence of polytopes $P = P_1, \ldots, P_s = \emptyset$ such that P_i is obtained from P_{i-1} by including an inequality which can be deduced from the inequalities of P_{i-1} by one of the two rules of Cutting Planes. In integer programming, we obtain P_i from P_{i-1} by a *Chvátal-Gomory cut*.

Theorem 7. *Stabbing Planes polynomially simulates Cutting Planes.*

Proof. We will say that a polytope P' can be deduced from P by Stabbing Planes if there is a query $(ax \leq b - 1,\ ax \geq b)$ such that $P \cap \{x : ax \leq b - 1\} = \emptyset$ and $P \cap \{x : ax \geq b\} \subseteq P'$.

Let P be an unsatisfiable polytope and $P = P_1, \ldots, P_s = \emptyset$ be a Cutting Planes refutation. To prove the theorem, we show that for each $i \in [s-1]$, $P_{i+1} = P_i \cap \{x : ax \geq b\}$ can be deduced from from P_i in Stabbing Planes by the query $(ax \leq b-1,\ ax \geq b)$. It remains to show that $P_i \cap \{x : ax \leq b - 1\} = \emptyset$. There are two cases, depending on the rule used to derive $ax \geq b$ from P_i.

- If $ax \geq b$ was derived by a conic combination of inequalities belonging to P_i, then $P_i \cap \{ax \leq b-1\} = \emptyset$ can be witnessed by adding the conic combination equalling $ax \geq b$ together with $ax \leq b - 1$ to deduce $0 \geq 1$.
- Otherwise, $ax \geq b$ is derived by division, i.e., it is $(c/t)x \geq \lceil c/t \rceil$ for some integer $t \geq 0$. Then $\lceil d/t \rceil \geq d/t$ and so $0 \geq 1$ is a conic combination of $(c/t)x \leq \lceil d/t \rceil - 1$ and $cx \geq d$, witnessing that $P \cap \{x : (c/t)x \leq \lceil d/t \rceil - 1\} = \emptyset$. □

To see that Stabbing Planes can simulate Cutting Planes, we view each inequality

4.2 Towards a Topology Preserving Simulation of Cutting Planes

An artifact of both the simulations of CP by variants of SP and of SP* by CP is that they are far from being *depth-preserving*; they convert shallow proofs into ones that are *extremely* deep. In this section, we explore whether this explosion in depth is inherent to the simulation of CP by SP. While we are unable to conclusively resolve this question — indeed, at this time the only technique for proving super-logarithmic depth lower bounds on SP works equally well for CP — we provide a number of depth-preserving simulations of subsystems of CP.

To motivate our results, we will take a detour and discuss the relationship between SP, CP and real communication protocols. Presently, almost all known lower bounds for CP are obtained by studying the communication complexity of the *false clause search problem* (defined in Section 2). For instance, it is known that:

- A depth d CP refutation yields a d-round real communication protocol for the associated false clause search problem.
- A size s treeCP refutation yields a real communication protocol $O(\log s)$-round real communication protocol for the associated false clause search problem.

- A size s and space ℓ CP refutation yields a $O(\ell \log s)$-round real communication protocol for the associated false clause search problem.
- A size s CP proof yields a dag-like real communication protocol for the associated false clause search problem.

All of these results have been used to derive strong lower bounds on Cutting Planes by proving the corresponding lower bound against the false clause search problem [12, 31, 36, 48, 54, 67]. Furthermore, this technique applies even to the stronger *semantic* CP system, as all one needs to exploit is that the lines are linear inequalities, rather than expoiting some weakness of the deduction rules. However, this strength also illustrates a weakness of current techniques, as once the lines of a proof system become expressive enough, proof techniques which work equally well for semantic proof systems 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 each of the simulation results above still hold if we replace real communication protocols with SP refutations, which are syntactic objects. That is, we show

(i) A depth d CP refutation yields a depth $2d$ SP refutation.

(ii) A size s treeCP refutation yields a size $O(s)$ and depth $O(\log s)$ SP refutation.

(iii) A size s and space ℓ CP refutation yields a size $O(2^\ell s)$ and depth $O(\ell \log s)$ SP refutation.

(iv) A size s CP refutation yields a size $O(s)$ SP refutation

4.2.1 Simulating CP Depth

First, we exhibit a depth-preserving simulation of CP by SP, which establishes (i). Furthermore, if the proof is *tree-like* then this simulation simultaneously preserves the size.

Theorem 8. $\mathsf{depth}_{\mathsf{CP}}(F) \geq 2 \cdot \mathsf{depth}_{\mathsf{SP}}(F)$. *Moreover, for any* treeCP *refutation of depth* d *and size* s *there is an* SP *refutation of depth* $2d$ *and size* $O(s)$.

Proof. It is sufficient to prove the "moreover" part of the statement, since, by recursive doubling, any CP refutation can be converted into a treeCP refutation where the depth remains the same.

Fix a treeCP refutation of size s and depth d, and let G be the its underlying tree. We will construct an SP refutation of the same system of linear inequalities by proceeding from the root of G to the leaves. In the process, we will keep track of a subtree T of G, which we have left to simulate, and an associated "current" node v of the SP refutation that we are constructing. Along the way, the following invariant will be maintained: at every recursive step (T, v) with $T \neq G$, if the root of T is labelled with the inequality $ax \geq b$, then the edge leading to v in the SP refutation is labelled with $ax \leq b - 1$.

Initially, $T = G$ and the SP refutation contains only a single root node v. Consider a recursive step (T, v). We break into cases based on which rule was used to derive the root of T.

- *Conic Combination.* Suppose that the root of T is labelled with an inequality $\lambda(a + c)x \geq \lambda(b + d)$ which was derived as a conic combination of $ax \geq b$ and $cx \geq d$. At the current node v in the SP refutation, query $(ax \leq b - 1,\ ax \geq b)$. On the branch labelled with $ax \geq b$, query $(cx \leq d - 1,\ cx \geq d)$. This sequence of queries results in three leaf nodes; see Figure 5. Let the leaf of the branch labelled with $ax \leq b - 1$ be ℓ_1 and let T_1 be the subtree rooted at the child of the root of T labelled with $ax \geq b$; recurse on (T_1, ℓ_1). Similarly, for the leaf ℓ_2 of the branch labelled with $cx \leq d - 1$, let T_2 be the sub-tree rooted at the child of the root of T labelled with $cx \geq d$, and recurse on (T_2, ℓ_2).

  For the final leaf, obtained by traversing the edges labelled with $ax \geq b$ and $cx \geq d$, we can derive $0 \geq 1$. To see this, first observe that if $T = G$ (i.e. the base case) then the root node of T is labelled with $0 \geq 1$ and $ax \geq b$ and $cx \geq d$ are the premises used to derive it by a conic combination. In this case, we can derive

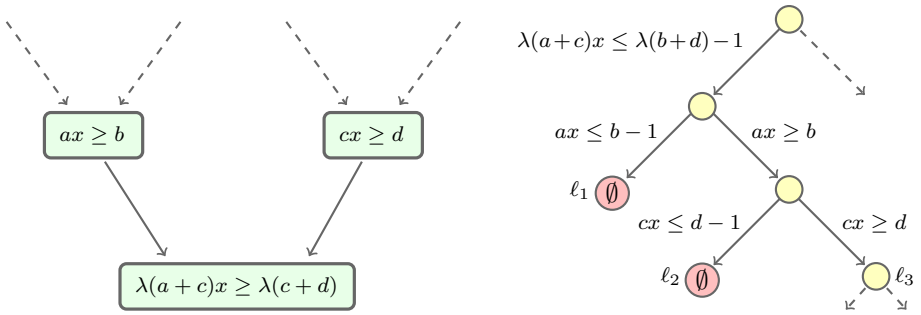

Figure 5: A treeR(CP) refutation invoking the conic combination rule (left) and the corresponding partial SP refutation (right).

$0 \geq 1$ by the same conic combination in SP. Otherwise, by the invariant, the edge leading to v is labelled with the inequality $\lambda(a+c)x \leq \lambda(b+d)-1$. Therefore, a conic combination of this inequality with $ax \geq b$ and $cx \geq d$ yields $0 \geq 1$.

- *Division.* If the root of T is labelled with an inequality $ax \geq \lceil b/\delta \rceil$ obtained by division from $\delta ax \geq b$, then query $(\delta ax \leq b-1,\ \delta ax \geq b)$. At the leaf ℓ_1 corresponding to the edge $\delta ax \leq b-1$, let T_1 be the subtree of T rooted at the child of the root of T and recurse on (T_1, ℓ_1). At the leaf corresponding to the edge $\delta ax \geq b$ we derive $0 \geq 1$ by a conic combination with $ax \leq \lceil b/\delta \rceil - 1$, which we have already deduced by the invariant. To see this, observe that $b - \delta(\lceil b/\delta \rceil - 1) > 0$

- *Axiom.* If T is a single node — a leaf of the treeCP refutation labelled with some initial inequality $ax \geq b$ of the system that it is refuting — then, by the invariant, we have already deduced $ax \leq b-1$ and this can be added to $ax \geq b$ to derive $0 \geq 1$.

To see that the SP refutation that we have constructed has depth at most twice that of the treeCP refutation, observe that conic combinations are the only inference rule of CP for which this construction requires depth 2 to simulate, while all other rules require depth 1.

To measure the size, note that every CP rule with a single premise is simulated in SP by a single query, where one of the outgoing edges of that query is immediately labelled with $0 \geq 1$. Each rule with two premises is simulated by two queries in the SP refutation, where one of the three outgoing edges is labelled with $0 \geq 1$. Therefore, the size of the SP refutation is $O(s)$. □

4.2.2 Balancing treeCP Proofs into SP

A proof system can be *balanced* if any proof of size s implies one of simultaneous size $\mathsf{poly}(s)$ depth $\mathsf{polylog}(s)$. While it is known that treeCP refutations cannot be balanced, we show next that if we permit the resulting refutation to be in SP, then we can balance. This establishes (ii).

Theorem 9. *Any size s treeCP refutation of an unsatisfiable system of linear inequalities $Ax \geq b$ implies a size $O(s)$ and depth $O(\log s)$ SP refutation of $Ax \geq b$.*

Proof. Consider a treeCP refutation of $Ax \geq b$ and let T be its corresponding tree. As well, let $|T|$ denote the number of nodes in T. We will construct the SP refutation recursively; at each step we will keep track of a current node u in the SP proof we are constructing. The base case is when $|T| = O(1)$, in which case we can use Theorem 8 to create an SP refutation of $Ax \geq b$ satisfying these properties, and append it to u.

For the recursive step, observe that because the tree has fanin at most 2, there exists a node v in T such that the subtree T_v rooted at v satisfies $|T|/3 \leq |T_v| \leq 2|T|/3$. Let $cx \geq d$ be the line corresponding to v. At node u in the SP proof, query $(cx \leq d-1,\ cx \geq d)$. Let u_1 (resp. u_2) be the child of u obtained by following the edge labelled with $cx \leq d-1$ (resp. $cx \geq d$); see Figure 6. We recurse as follows:

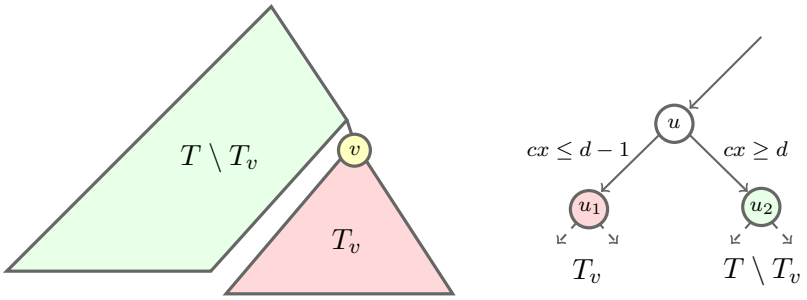

Figure 6: A decomposition of treeR(CP) tree T into T_v and $T \setminus T_v$ (left) and the corresponding partial SP refutation (right).

- At u_1; Observe that the sub-proof T_v is a treeCP derivation of the inequality $cx \geq d$. Because we have deduced $cx \leq d-1$ on the path to u_1, if we also deduce $cx \geq d$ then this is sufficient to derive $0 \geq 1$. Therefore, at u_1 we recurse on the treeCP derivation T_v.
- At u_2: Observe that the sub-proof $T \setminus T_v$ is a treeCP refutation of $Ax \geq b$ where we have assumed $cx \geq d$ as an axiom. Therefore, at u_2 we recursively construct an SP refutation of the set of inequalities $\{Ax \geq b, cx \geq d\}$ using tree $T \setminus T_v$.

The size of the treeCP refutation is clearly preserved. Observe that the depth of the resulting SP refutation becomes logarithmic in s, since we are reducing the size of the proof to be simulated by a constant factor on each branch of a query. □

4.2.3 Balancing Low-Space CP Proofs into SP

Next,we show how to balance CP proofs into SP, provided the *space* at each step of the proof is bounded.

The space for a proof system models the amount of information that must be remembered at each state in the nondeterministic Turing machine that underlies a proof system. To capture this, we redefine a Cutting Planes refutation of a system of linear inequalities $Ax \geq b$ as a sequence of configurations $C_1, \ldots, C_s$ where a configuration C_i is a set of integer linear inequalities satisfying the following conditions: (i) $C_1 = \emptyset$, (ii) C_s contains the inequality $0 \geq 1$, (iii) each configuration C_t follows from C_{t-1} by removing any number of inequalities and including an inequality which was derived from inequalities in C_{t-1} by one of the rules of CP or an initial inequality belonging to $Ax \geq b$. The *line space* of a refutation is $\max_{i \in [s]} |C_i|$, the maximum number of inequalities in any configuration.

Theorem 10. *For any* CP *refutation of size s and line space ℓ of a system of linear inequalities $Ax \geq b$ there is an* SP *refutation of depth $O(\ell \log s)$ and size $O(s \cdot 2^{\ell})$.*

This implies that strong depth lower bounds on SP proofs can lead to size-depth tradeoffs for CP.

Proof. Fix a Cutting Planes refutation $C_1, \ldots, C_\ell$ where $|C_i| \leq \ell$ for all $i \in [s]$. The theorem will follow by taking $i = s$ in the following claim.

Claim. For any $i \in [s]$ there exists an SP tree of depth $2\ell \log i$ such that every root-to-leaf path ends in a leaf labelled with $0 \geq 1$, except for one, along which we have deduced all of the inequalities in C_i.

Proof of Claim. It remains to prove the claim. If C_i contains only a single inequality $a_i x \geq b_i$ and it belongs to $Ax \geq b$, then take the tree to be the one corresponding to the SP query $(a_i x \leq b_i - 1,\ a_i x \geq b_i)$. Otherwise, the SP tree begins with a complete binary tree in which every inequality in $C_{\lfloor i/2 \rfloor}$ is queried. Exactly one path in this tree is labelled with the inequalities in $C_{\lfloor i/2 \rfloor}$, and the remaining paths contain the integer negation (i.e., $cx \leq d-1$) of at least one inequality $cx \geq d$ in $C_{\lfloor i/2 \rfloor}$. We consider these two cases separately.

In the case that a path contains a negation of a line from $C_{\lfloor i/2 \rfloor}$, we attach to its leaf the SP tree we obtain recursively by running our construction on $C_1, \ldots, C_{\lfloor i/2 \rfloor}$. The leaves of the resulting tree are all labelled with

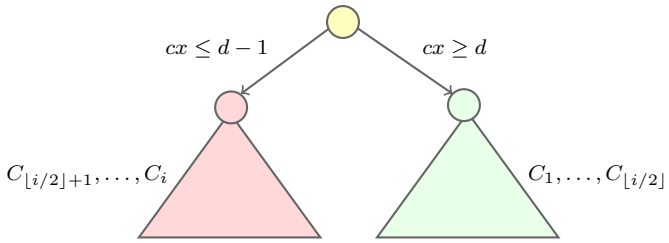

Figure 7: The SP tree corresponding to a configuration $C_i = \{cx \ge d\}$.

$0 \ge 1$, except for one. By construction, at the leaf not labelled with $0 \ge 1$ we have deduced all inequalities in $C_{\lfloor i/2 \rfloor}$. Since we attached this tree to a path along which we had deduced the negation of a line in $C_{\lfloor i/2 \rfloor}$, we can label this leaf with a conic combination of these inequalities equalling $0 \ge 1$. The overall depth in this case is ℓ for the initial tree and $2\ell \log(\lfloor i/2 \rfloor)$ for the tree obtained recursively. Altogether, $\ell + 2\ell \log(\lfloor i/2 \rfloor) \le \ell(1 + 2\log(i) - 2) \le 2\ell \log i$.

For the path labelled with the inequalities in $C_{\lfloor i/2 \rfloor}$, note that $C_{\lfloor i/2 \rfloor+1}, \ldots, C_i$ can viewed as configurations of a refutation of the original inequalities $Ax \ge b$ together with the inequalities in $C_{\lfloor i/2 \rfloor}$. At the leaf of this path we have deduced all inequalities in $C_{\lfloor i/2 \rfloor}$. Thus, we can apply the recursive construction to $C_{\lfloor i/2 \rfloor+1}, \ldots, C_i$ to refute this leaf. The overall depth is $\ell + 2\ell \log(\lfloor i/2 \rfloor) \le \ell(1 + 2\log(i+2) - 2) \le 2\ell \log i$. □

4.3 Simulating non-CG Cuts

So far, we have focused on the relationship between Stabbing Planes and Chvátal-Gomory cutting planes. In this section we discuss the relationship between SP and other popular types of cutting planes. First, we cover the result of Basu et al. [8] which shows that SP can simulate *split cuts*. Split cuts, which were introduced by Cook et al. [22], and form one of the most popular classes of cutting planes in practical integer linear programming. Recall that an inequality $ax \ge b$ is valid for a polytope P if for every $x^* \in P$, $ax^* \ge b$.

Split Cut. A *split cut* for a polytope P is any integer-linear inequality $ax \ge b$ for which there exists a *witnessing* pair $c \in \mathbb{Z}^n$ and $d \in \mathbb{Z}$ such that $ax \ge b$ is valid for both $P \cap \{x \in \mathbb{R}^n : cx \le d-1\}$ and $P \cap \{x \in \mathbb{R}^n : cx \ge d\}$.

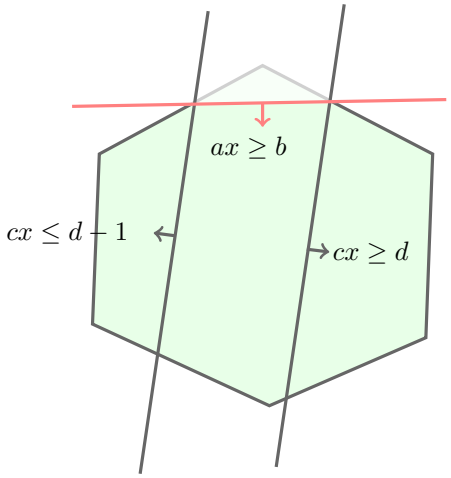

Figure 8: A split cut $ax \ge b$ witnessed by $(cx \le d-1,\ cx \ge d)$ on the polytope in green.

Split cuts are known to be equivalent to *mixed-integer rounding* (MIR) cuts [63] and *Gomory mixed integer* cuts [40], and generalize *lift-and-project* cuts [6]. As well, Dash [28] gave an example on which split cuts are exponentially separated from Chvátal-Gomory cuts and lift-and-project cuts. Basu et al. [8] showed that split cuts can be simulated in Stabbing Planes. For completeness, we include a proof of this.

Lemma 11 ([8]). *Let P be a polytope and let $P' = P \cap \{ax \geq b\}$ be obtained by a split cut from P. Then, there is an* SP *tree of size $O(1)$ beginning from P such that every leaf is empty, except for one whose corresponding polytope is P'.*

Proof. We simulate the deduction of P' from P in Stabbing Planes as follows:

(i) Query $(ax \leq b-1,\ ax \geq b)$

(ii) On the branch labelled with $ax \leq b-1$, query $(cx \leq d-1,\ cx \geq b)$. Observe that because $ax \geq b$ was valid for both $P \cap \{cx \geq d\}$ and $P \cap \{cx \leq d-1\}$, it follows that both $P \cap \{cx \geq d\} \cap \{ax \leq b-1\}$ and $P \cap \{cx \geq d\} \cap \{ax \leq b-1\}$ are empty.

Therefore, the only non-empty leaf is the one corresponding to P'. □

Dash [28] studied split cuts as a proof system, and showed that the lower bound of Pudlak [67] could be extended to prove exponential lower bounds on the length of split cut proofs. A split cut refutation of a system of integer linear inequalities $Ax \geq b$ (representing a polytope P) is a sequence of inequalities $c_1 x \geq d_1, \ldots, c_s x \geq d_s = 0 \geq 1$ such that $c_i x \geq d_i$ is a split cut for the polytope $P \cap \{c_j x \geq d_j\}_{j<i}$. The following is immediate corollary of Lemma 4.3.

Corollary 12. *Stabbing Planes polynomially simulates split cut proofs.*

Proof. To simulate any split cut refutation $c_1 x \geq b_1, \ldots, c_s x \geq b_s$, we simply simulate each cut inductively using Lemma 4.3. □

Finally, we note that there exist cutting planes that cannot be efficiently simulated by SP. This is witnessed by the fact that SP cannot polynomially simulate *semantic* CP [33]. A concrete example of a type of cutting plane that SP likely cannot simulate are the *matrix cuts* of Lovász and Schrijver [60]. As we describe in Section 5 CP, and therefore, SP^* cannot quasi-polynomially simulate the Lovasz-Schrijver proof system. However, whether this holds for SP with unbounded coefficients remains an interesting question.

5 Relationship Between Stabbing Planes and Other Proof Systems

Having explored in depth the relationship between Cutting Planes and Stabbing Planes, we now describe how Stabbing Planes relates to other proof systems. A summary of these relationships can be seen in Figure 2.

Let us first note some of the separations that have already been established.

- Lower bounds for unsatisfiable systems of linear equations over finite fields, which are known for *Nullstellensatz* [44], the *Polynomial Calculus* [15], *Sum-of-Squares* [45, 70], AC^0*-Frege* [11, 38, 47, 66], rule out the possibility of these systems simulating CP.

- Göös et al. [42] gave an exponential separation between *Nullstellensatz* and Cutting Planes by observing that, for any unsatisfiable system of linear equations F, composing with the m-bit *index gadget* can only increase the degree of refuting F in Nullstellensatz by $O(\log m)$. On the other hand, Garg et al. [39] showed that composing any function which requires resolution refutations of *width* w when composed with the index gadget requires Cutting Planes proofs of size $n^{\Omega(w)}$. Thus, any function which requires large resolution width but small Nullstellensatz degree provides such a separation.

- Semantic CP is not polynomially verifiable, and therefore, assuming $\mathsf{P} \neq \mathsf{NP}$, no propositional proof system can simulate it. Indeed, Filmus, Hrubeš, and Lauria observed that it has $O(1)$ size refutations of unsatisfiable instances of the NP-complete subset sum problem.

We establish the remaining simulations and separations in Figure 2 next.

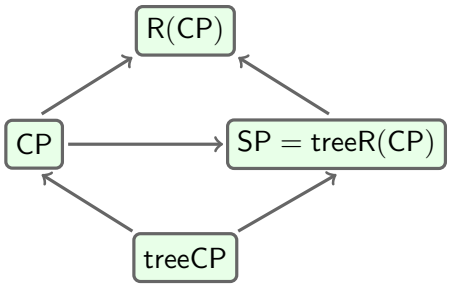

Figure 9: Relationships between R(CP), CP, and their tree-like variants. An arrow from proof system P_1 to P_2 indicates that P_2 can polynomially simulate P_1.

5.1 Equivalence Between Stabbing Planes and treelike R(CP)

The Resolution over Cutting Planes (R(CP)) proof system was introduced by Krajíček [54] 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 previously derived disjunctions and either apply a Cutting Planes rule to a single inequality in the disjunctions, or apply a resolution-style "cut".

Resolution over Cutting Planes. An R(CP) *proof* of a disjunction Γ_s from a system of integer-linear inequalities $Ax \geq b$ is a sequence of disjunctions $P = \{\Gamma_i\}_{i\in[s]}$ such that each Γ_i is a disjunction which is either an inequality from $Ax \geq b$ or was derived from earlier disjunctions by one of the following deduction rules:

- *Conic Combination*. From $(ax \geq b) \vee \Gamma$ and $(cx \geq d) \vee \Gamma$ deduce $(\lambda(a+c)x \geq \lambda(b+d)) \vee \Gamma$ for any non-negative integer λ.
- *Division*. From $(ax \geq b) \vee \Gamma$ and integer δ dividing each entry of a, deduce $((a/\delta)x \geq \lceil b/\delta \rceil) \vee \Gamma$.
- *Cut*. From $(ax \geq b) \vee \Gamma$ and $(ax \leq b-1) \vee \Gamma$ derive Γ.
- *Weakening*. From Γ deduce $\Gamma \vee (ax \geq b)$
- *Axiom Introduction*. Deduce $(ax \geq b) \vee (ax \leq b-1)$ for any integer-linear inequality $ax \geq b$.
- *Elimination*. From $(0 \geq 1) \vee \Gamma$ deduce Γ.

The *size* of a proof is the number of disjunctions s in the proof and the *width* of the proof is the maximal number of inequalities in any disjunction in the proof. An R(CP) *refutation* of $Ax \geq b$ is a proof of the empty clause Λ from $Ax \geq b$. The proof system treeR(CP) is the tree-like restriction of R(CP) in which the underlying implication graph is required to be a tree.

The main result of this sub-section is that SP is polynomially equivalent to treeR(CP).

Theorem 2. *The proof systems* SP *and* treeR(CP) *are polynomially equivalent.*

Even though SP turns out to be equivalent to a system already in the literature, this new perspective has already shown to be useful: none of aforementioned results were known for treeR(CP).

We will prove Theorem 2 in two parts.

Claim 13. Let $Ax \geq b$ be an unsatisfiable system of m integer-linear inequalities. Any size s and depth d SP refutation implies a treeR(CP) refutation of size $O(s(d^2 + dm))$ and width $d+1$.

Proof. Consider an SP refutation of $Ax \geq b$ of size s and depth d. Fix any root-to-leaf path p in the refutation and let $c_1x \geq d_1, \ldots, c_tx \geq d_t$ be the sequence of linear inequalities labelling p. We will first show how to derive the clause

$$(c_1x \leq d_1 - 1) \vee \ldots \vee (c_tx \leq d_t - 1) \tag{1}$$

in treeR(CP). For every $i \in [t]$, using *axiom introduction*, introduce $(c_i x \leq d_i - 1) \vee (c_i x \geq d_i)$ and *weaken* it to obtain

$$(c_i x \geq d_i) \vee (c_1 x \leq d_1 - 1) \vee \ldots \vee (c_t x \leq d_t - 1). \tag{2}$$

As well, weaken each initial inequality $a_i x \geq b_i$ in $Ax \geq b$ to

$$(a_i x \geq b_i) \vee (c_1 x \leq d_1 - 1) \vee \ldots \vee (c_t x \leq d_t - 1) \tag{3}$$

Because p is a root-to-leaf path in the SP proof, it is labelled with a conic combination of $Ax \geq b$ and $c_i x \geq d_i$ for every $i \in [t]$ equalling $0 \geq 1$. By taking this conic combination of the first inequalities of the lines in (2) and (3) we can deduce

$$(0 \geq 1) \vee (c_1 x \leq d_1 - 1) \vee \ldots \vee (c_t x \leq d_t - 1),$$

from which we can obtain (1) by *elimination*.

Repeat this process to deduce (1) for every root-to-leaf path in the SP proof. Applying the *cut* rule appropriately to these inequalities yields the empty clause. To see this, let p and p' be two root-to-leaf paths which differ only on their leaf nodes. Then, their corresponding inequalities (1) are of the form

$$(c_i x \geq d_i) \vee (c_1 x \leq d_1 - 1) \vee \ldots \vee (c_{t-1} x \leq d_{t-1} - 1) \vee (c_t x \leq d_t - 1),$$
$$(c_i x \geq d_i) \vee (c_1 x \leq d_1 - 1) \vee \ldots \vee (c_{t-1} x \leq d_{t-1} - 1) \vee (c_t x \geq d_t).$$

That is, they differ in their final inequality. Applying the *cut* rule, we can deduce

$$(c_i x \geq d_i) \vee (c_1 x \leq d_1 - 1) \vee \ldots \vee (c_{t-1} x \leq d_{t-1} - 1).$$

Therefore, by repeating this process we can derive the empty clause.

Each deduction of a clause (1) can be done in size $O(t^2 + tm + t + m) = O(d^2 + dm)$ and has width at most $d + 1$. Thus, the size of the proof is at most $O(s(d^2 + dm))$. □

We now prove the converse.

Claim 14. Let $Ax \geq b$ be an unsatisfiable system of m integer-linear inequalities. If there is a treeR(CP) proof of the line $(a_1 x \leq b_1 - 1) \vee \ldots \vee (a_m x \leq b_m - 1)$, where $a_i x \geq b_i$ is the ith row of $Ax \geq b$, of size s and depth d then there is an SP refutation of $Ax \geq b$ of size $O(s)$ and depth $2d$.

Proof. Fix such a treeR(CP) proof of the disjunction. For any clause $\Gamma = (c_1 x \geq d_1) \vee \ldots (c_m x \geq d_m)$, we will denote by $\neg\Gamma$ the set of inequalities $\{c_1 x \leq d_1 - 1, \ldots, c_m x \leq d_m - 1\}$. We will construct the SP refutation by structural induction, beginning at the leaves of the refutation and proceeding towards the root. At each line Γ in the proof, deduced from children Γ_1 and Γ_2, we will assume that we have constructed SP refutations $\neg\Gamma_1$ and $\neg\Gamma_2$ and use them to construct an SP refutation of $\neg\Gamma$.

First, consider a leaf of the proof which, by definition, is an *axiom introduction* of $(cx \leq d - 1) \vee (cx \geq d)$ for some arbitrary integer-linear inequality $cx \geq d$. We can construct an SP refutation of $(cx \geq d) \wedge (cx \leq d - 1)$ by querying $(cx \leq d - 1,\ cx \geq d)$, and then labelling each leaves with the appropriate conic combination equalling $0 \geq 1$.

Now, let Γ be some line in the proof which was derived from earlier lines $\{\Gamma_i\}$, and suppose that we have constructed SP refutations of $\{\neg\Gamma_i\}$. To construct a refutation of $\neg\Gamma$, we break into cases based on the rule used to derive Γ.

- *Conic combination.* Let $\Gamma := (\lambda(c_1 + c_2)x \geq \lambda(d_1 + d_2)) \vee \Delta$, let $\Gamma_1 := (c_1 x \geq d_1) \vee \Delta$, and let $\Gamma_2 := (c_2 x \geq d_2) \vee \Delta$. We construct an SP refutation of $\neg\Gamma$ by first querying $(c_1 x \leq d_1 - 1,\ c_1 x \geq d_1)$. On the branch labelled with $c_1 x \geq d_1$, apply the SP refutation of $\neg\Gamma_1$. On the branch labelled with $c_1 x \geq d_1$, query $(c_2 x \leq d_2 - 1,\ c_2 x \geq d_2)$, and use the refutation of $\neg\Gamma_2$ to refute the branch labelled with $c_2 x \geq d_2$. On the remaining branch, where we have deduced $c_1 x \leq d_1 - 1$ and $c_2 x \leq d_2 - 1$, we have that $0 \geq 1$ is a conic combination with $\lambda(c_1 + c_2)x \geq \lambda(d_1 + d_2)$.

- *Division.* Let $\Gamma := ((c/\delta)x \geq \lceil d/\delta \rceil) \vee \Delta$ and let $\Gamma_1 = (\delta cx \geq d) \vee \Delta$. Query $(\delta cx \leq d-1,\ \delta cx \geq d)$. On the branch labelled with $cx \leq d-1$ we can use the refutation of $\neg\Gamma_1$. On the branch labelled with $cx \geq d$, it is enough to observe that the intersection of $cx \geq d$ and $((c/\delta)x \leq \lceil d/\delta \rceil - 1)$, provided by $\neg\Gamma$, is empty.
- *Cut.* Suppose $\Gamma := \Delta$ was derived by cutting on $\Gamma_1 := (cx \geq d) \vee \Delta$ and $\Gamma_2 := (cx \leq d-1) \vee \Delta$. Query $(cx \leq d-1,\ cx \geq d)$. On the branch labelled with $cx \leq d-1$ apply the refutation of $\neg\Gamma_1$, and on the branch labelled with $cx \geq d$ use the refutation of $\neg\Gamma_2$.
- *Weakening.* If $\Gamma := (cx \geq d) \vee \Delta)$ was derived by weakening $\Gamma_1 := \Delta$, then query $(cx \leq d-1,\ cx \geq d)$. On the branch labelled with $cx \leq d-1$ use the refutation of $\neg\Gamma_1$, and the branch labelled with $cx \geq d$ we can deduce $0 \geq 1$ by adding this inequality to $cx \leq d-1$ with is an inequality of $\neg\Gamma$.

Simulating each rule requires at most two queries, of which at most two of the children are not immediately the empty polytope. Therefore, the size of the resulting tree is at most $2s$ and the depth is at most $2d$. □

5.2 Stabbing Planes Simulates Tree-like DNF Resolution

Next, we show how to simulate the k-DNF resolution proof systems by variants of Stabbing Planes.

k-DNF Resolution. A $\mathsf{Res}(k)$ refutation of a CNF formula F is a sequence of k-DNF formulas $P = \{\Gamma_i\}_{i\in[s]}$ such that Γ_s is the empty clause Λ, and each Γ_i is either a clause of F or was derived from earlier DNFs by one of the following deduction rules, where a literal ℓ_i is either x_i or $\neg x_i$:

- *Cut*. From k-DNFs $A \vee (\wedge_{i\in S}\ell_i)$ and $B \vee (\vee_{i\in S}\neg\ell)$ deduce $A \vee B$.
- *Weakening*. From a k-DNF A deduce $A \vee \ell$ for any literal ℓ.
- *$\wedge$-Introduction*. From $\{A \vee \ell_i\}_{i\in S}$ deduce $A \vee (\wedge_{i\in S}\ell_i)$.
- *$\wedge$-Elimination*. From $A \vee (\wedge_{i\in S}\ell_i)$ deduce $A \vee \ell_i$ for any $i \in S$.

A refutation is *tree-like* if every deduced inequality is used at most once in the refutation (i.e., the underlying implication graph is a tree). The proof system which produces only treelike $\mathsf{Res}(k)$ refutations is denoted $\mathsf{treeRes}(k)$.

Theorem 15. *For any integer $k \geq 1$, any $\mathsf{Res}(k)$ refutation of size s implies an $\mathsf{R(CP)}$ refutation of size $O(ks)$. Similarly, any $\mathsf{treeRes}(k)$ refutation of size s implies an SP refutation of size $O(ks)$.*

The proof will follow by a straightforward application of the following claim.

Claim 16. From any disjunction $\bigvee_{i\in S}(x_i \geq 1) \vee \bigvee_{j\in T}(-x_i \geq 0)$, together with inequalities $x_i \geq 0$ and $x_i \leq 1$ for every $i \in [n]$, there is a size $O(|S| + |T|)$ $\mathsf{treeR(CP)}$ derivation of $\sum_{i\in S} x_i + \sum_{j\in T}(1 - x_j) \geq 1$.

Proof. For every $v \in T \cup S$, derive $\sum_{i\in T\setminus\{v\}} x_i + \sum_{j\in S\setminus\{v\}}(1-x_j) \geq 0$ by adding together the inequalities $x_i \geq 0$ and $x_i \leq 1$. For $v \in T \cup S$ add the corresponding inequality to the disjunction in $\bigvee_{i\in S}(x_i \geq 1) \vee \bigvee_{j\in T}(-x_j \geq 0)$ containing the variable v. The result is the disjunction

$$\bigvee_{S\cup T}\Big(\sum_{i\in S} x_i + \sum_{j\in T}(1-x_j) \geq 1\Big),$$

which is the inequality $\sum_{i\in S} x_i + \sum_{j\in T}(1-x_j) \geq 1$. □

Proof of Theorem 15. We will show that $\mathsf{R(CP)}$ can simulate $\mathsf{Res}(k)$. That SP simulates $\mathsf{treeRes}(k)$ will follow by observing that the same proof also shows a simulation of $\mathsf{treeRes}(k)$ by $\mathsf{treeR(CP)}$, and then applying Theorem 2.

Let $\{\Gamma_i\}_{i\in[s]}$ be a $\mathsf{Res}(k)$ refutation of a CNF formula F, and note that the encoding of F as a system of inequalities (recalled in Section 2) includes $x_i \geq 0$ and $x_i \leq 1$ for every $i \in [n]$. We will encode each disjunction $\Gamma := \Delta_1 \vee \ldots \vee \Delta_t$ as follows: each $\Delta := (\wedge_{i\in S}x_i) \wedge (\wedge_{j\in T}\neg x_j)$ is represented by the inequality $\sum_{i\in S}(x_i - 1) + \sum_{j\in T} -x_j \geq 0$; observe that both representations are satisfied by the same set of $\{0,1\}$-assignments. Let L_Γ be the encoding of Γ obtained by replacing each Δ_i by its encoding as an inequality.

It remains to show that $\mathsf{R(CP)}$ can simulate the deduction rules of $\mathsf{Res}(k)$.

- *Cut*. Suppose that $\Gamma := A \vee B$ be deduced by cutting on $\Gamma_1 := A \vee (\wedge_{i \in S} x_i) \wedge (\wedge_{j \in T} \neg x_j)$ and $\Gamma_2 := B \vee (\vee_{i \in S} \neg x_i) \vee (\vee_{j \in T} x_j)$. As well, suppose that we have already deduced the corresponding lines $L_{\Gamma_1} := L_A \vee (\sum_{i \in S}(x_i - 1) + \sum_{j \in T} -x_j \geq 0)$ and $L_{\Gamma_2} := L_B \vee \bigvee_{i \in S}(-x_i \geq 0) \vee \bigvee_{j \in T}(x_j \geq 1)$. By Claim 16, $\mathsf{R(CP)}$ can reencode L_{Γ_2} as $L_B \vee (\sum_{i \in S}(1 - x_i) + \sum_{j \in T} x_j \geq 1)$, which when added to L_{Γ_1} gives $L_A \vee L_B \vee (0 \geq 1)$, which is L_Γ.
- *Weakening*. This is already a rule of $\mathsf{R(CP)}$.
- $\wedge$*-Introduction*. If $\Gamma := A \vee (\wedge_{i \in S} x_i) \wedge (\wedge_{j \in T} \neg x_j)$ was deduced from $\{A \vee x_i\}_{i \in S}$ and $\{A \vee \neg x_j\}_{j \in T}$ and we have already deduced $L_{\Gamma_i} := L_A \vee (x_i \geq 1)$ and $L_{\Gamma_j} := L_A \vee (-x_j \geq 0)$ for all $i \in S$ and $j \in T$. Then L_Γ can be deduced by adding together all of the L_{Γ_i} and L_{Γ_j}.
- $\wedge$*-Elimination*. If $\Gamma = A \vee x_i$ was deduced from $\Gamma_1 := A \vee (\wedge_{j \in S} x_j) \wedge (\wedge_{t \in T} x_t)$, then L_Γ can be deduced from $L_{\Gamma_1} := A \vee (\sum_{j \in S}(x_j - 1) + \sum_{t \in T} -x_t \geq 0)$ by adding the inequalities $x_j \leq 1$ for every $j \in S \setminus \{i\}$ and $x_t \geq 0$ for every $t \in T$. A similar argument holds if x_i is negated.

□

Atserias, Bonet, and Estaban [4] gave polynomial-size proofs of the *clique-coclique* formulas in $\mathsf{treeRes}(k)$, for cliques of size $\Omega(\sqrt{n})$ and cocliques of size $o(\log^2 n)$. For this range of parameters, quasi-polynomial size lower bounds are known [67]. This rules out the possibility of a *polynomial* simulation of $\mathsf{R(CP)}$ or $\mathsf{treeRes}(k)$ by Cutting Planes.

6 Lower Bounds on Stabbing Planes

Next, we tackle the problem of proving lower bounds on Stabbing Planes proofs. First, we show that near-maximal depth lower bounds on unrestricted Stabbing Planes proofs can be obtained by a straightforward reduction to communication complexity. Next, while we are unable to prove unrestricted size lower bounds, we explain why current techniques that would attempt to leverage the depth lower bounds fail. In doing so, we show that real communication protocols cannot be balanced by establishing the first superlogarithmic lower bound on the real communication complexity of the set disjointness function.

First, we recall some standard models of communication and previous lower bounds on depth via communication complexity. In proving lower bounds on proof complexity it has been fruitful to study the following associated search problem, introduced by Lovász et al. [58].

False Clause Search Problem. Let $F = C_1 \wedge \ldots \wedge C_m$ be a CNF formula and let (X, Y) be any partition of its variables. The associated false clause search problem $\mathsf{Search}_F^{(X,Y)} \subseteq \{0,1\}^{|X|} \times \{0,1\}^{|Y|} \times [m]$ is defined as $(x, y, i) \in \mathsf{Search}_F^{(X,Y)}$ if and only if $C_i(x, y) = 0$.

Our depth lower bounds are inspired by the approach of Impgliazzo et al. [49] who observed that small treelike Cutting Planes proofs implied short protocols in certain models of communication for solving the following false clause search problem.

Deterministic Communication. A deterministic communication protocol for a search problem $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$ consists of two players, Alice and Bob. They receive private inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ respectively, and their aim is to agree on some $o \in \mathcal{O}$ for which $(x, y, o) \in \mathcal{S}$. To do so, they are allowed to communicate by sending messages to each other (in the form of a single bit) according to some predetermined *protocol*. This can be modelled combinatorially: every step in the communication protocol is associated with *rectangle* of inputs $\mathcal{X}' \times \mathcal{Y}' \subseteq \mathcal{X} \times \mathcal{Y}$ consistent with the communication thus far; $\mathcal{X}'$ models what Bob knows about Alice's input, and $\mathcal{Y}'$ models what Alice knows about Bob's. If Alice communicates a bit, then this partitions $\mathcal{X}'$ into $\mathcal{X}'_0$ and $\mathcal{X}'_1$ corresponding to whether the bit Alice sent was 0 or 1. The communication ends when $\mathcal{X}' \times \mathcal{Y}'$ is *monochromatic*, meaning that there is some $o \in \mathcal{O}$ such that $(x, y, o) \in \mathcal{S}$ for every $(x, y) \in \mathcal{X}' \times \mathcal{Y}'$.

The *deterministic communication complexity* of computing $\mathcal{S}$ is the minimum number of bits communicated, or *rounds* of communication, needed to solve $\mathcal{S}$ on any input $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

Observe that for $x, y \in \{0,1\}^n$, any integer linear inequality $ax + by \geq d$ can be evaluated in $w = \log \|a\|_1 + \log \|b\|_1$ bits by Alice communicating ax to Bob, and Bob responding with by. Therefore, for example, a Cutting Planes refutation of a CNF formula F of depth d in which the size of the coefficients are at most 2^w implies a $O(dw)$-round communication protocol for solving $\mathsf{Search}_F^{(X,Y)}$ for any partition (X, Y) of the variables. By strengthening the model of communication, we can simulate arbitrary linear inequalities. Next, we define two models of communication which allow us to do this; the first is the standard model of randomized communication complexity.

Randomized Communication. A (bounded error) randomized communication protocol solving a search problem $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$ is a distribution over deterministic communication protocols such that for every $(x, y) \in \mathcal{X} \times \mathcal{Y}$, with probability at least $2/3$, the protocol outputs o for which $(x, y, o) \in \mathcal{S}$. The randomized communication complexity of $\mathcal{S}$ is the minimum number of rounds of any randomized protocol computing $\mathcal{S}$, where the number of rounds of a randomized protocol is the maximum number of rounds of any protocol with non-zero support in the distribution.

An alternative model, which more directly simulates arbitrarily linear inequalities, is the *real communication* model introduced by Krajíček [54].

Real Communication. In a *real communication protocol* for a search problem $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{O}$, the players Alice and Bob communicate via a "referee". In each round, Alice and Bob send real numbers r_A, r_B to the referee who responds with a single bit b which is 1 if $r_A > r_B$, and 0 otherwise. The *real communication complexity* of computing $\mathcal{S}$ is the number minimum number of rounds needed to communication needed to solve $\mathcal{S}$ on any input $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

6.1 Depth Lower Bounds

In this section we prove Theorem 3, which we restate next for convenience.

Theorem 3. *There exists a family of unsatisfiable CNF formulas $\{F_n\}$ for which any* SP *refutation requires depth* $\Omega(n/\log^2 n)$

Note that every unsatisfiable CNF formula has a refutation in depth n by simply querying $(x_i \leq 0,\ x_i \geq 1)$ for all $i \in [n]$. Therefore, this lower bound is tight up to a $\log^2 n$ factor.

The proof proceeds by showing that from any shallow SP refutation we can extract a short *randomized* or *real* communication protocol for the associated false clause search problem. The lower bound follows by appealing to known lower bounds on the communication complexity of this problem.

Lemma 17. *Let $Az \geq b$ be an unsatisfiable system of linear equations encoding a CNF formula F and let (X, Y) be any partition of the variables z. Every depth d* SP *refutation of $Az \geq b$ implies a $O(d \log n + \log^2 n)$-round randomized communication protocol and a $O(d + \log n)$-round real communication protocol for solving* $\mathsf{Search}_{X,Y}(F)$.

Proof. We will first present a general procedure for solving the false clause search problem and then show how to instantiate it in both models of communication.

Fix an SP refutation of $Az \geq b$. Let Alice be given a boolean assignment to X and Bob be given a boolean assignment to Y. To solve the search problem, they will follow the root-to-leaf path through the refutation, maintaining the invariant that their joint assignment (X, Y) satisfies all of the inequalities labelling the root to leaf path. Suppose that they have arrived at a node in the refutation corresponding to a query $(cz \leq d-1,\ cz \geq d)$. Observe that their joint assignment (X, Y) to z satisfies exactly one of these two inequalities. They will proceed down the path corresponding to the satisfied inequality, thus preserving their invariant.

Once they arrive at a leaf, they will use the conic combination of inequalities which evaluates to $0 \geq 1$ that labels it in order to search for an inequality of $Az \geq b$ (corresponding to a clause of F) which is falsified by (X, Y). Indeed, by the invariant, the only inequalities in this conic combination which could be falsified by (X, Y) are those belonging to $Az \geq b$, and a falsified inequality must exist because (X, Y) falsifies $0 \geq 1$. Let the conic combination be $\sum_{i \in [\ell]} \alpha_i c_i z \leq \sum_{i \in [\ell]} \alpha_i d$, where $\alpha_i \geq 0$. By Carathéodory's Theorem (point (ii) in Farkas' Lemma), we can

assume that $\ell \leq n+2$. To find a falsified inequality, we binary search over the conic combination: test whether $\sum_{i=1}^{\ell/2} \alpha_i c_i z \leq \sum_{i=1}^{\ell/2} \alpha_i d_i$ is falsified by (X, Y). If it is, recurse on it; otherwise, recurse on $\sum_{i=\ell/2+1}^{\ell} \alpha_i c_i z \leq \sum_{i=\ell/2+1}^{\ell} \alpha_i d_i$. Because $\ell \leq n+2$, this process terminates in $O(\log n)$ rounds having found an inequality belonging to $Az \geq b$ which is falsified by (X, Y).

To implement this procedure in communication, it remains to show that linear inequalities can be evaluated efficiently in each of the models.

- *Real communication*: this can be done in a single round of communication. If Alice and Bob want to evaluate $c_1 x + c_2 y \geq d$, then Alice can send $c_1 x$ to the referee and Bob can send $d - c_2 y$. The referee returns whether $c_1 x \geq d - c_2 y$.
- *Randomized communication*: this can be done in $O(\log n)$ rounds of communication by combining the following two results. The first is the $O(\log b)$ protocol of Nisan [64] for deciding a linear inequality representable in b bits. The second is a result due to Muroga [62] which states that for any linear inequality on n variables, there exists a linear inequality whose coefficients are represented in $O(n \log n)$ bits and which has the same output on points in $\{0,1\}^n$.

□

To establish Theorem 3, it remains to lower bound the communication complexity of $\mathsf{Search}(F)$. Strong lower bounds on the randomized communication complexity of the false clause search lower bound were proven by Göös and Pitassi [43]. In particular, Theorem 8.1 in [43] gives an unsatisfiable CNF formula F on $\mathsf{poly}(n)$ many clauses and partition (x, y) of the variables for which the randomized commutation complexity of $\mathsf{Search}_{x,y}(F)$ requires $\Omega(n/\log n)$ rounds. Together with Lemma 17, this establishes Theorem 3.

We remark that the formula provided by Göös and Pitassi is somewhat artificial. It is obtained by *lifting* the Tseitin formulas with a *versatile* gadget. By Theorem 6 we know that the Tseitin formulas have $O(\log^2 n)$-depth SP refutations, and therefore the hardness of these formulas of Göös and Pitassi is derived from the composition with this gadget. It remains an open problem to obtain strong lower bounds on the depth of SP refutations for more natural families of formulas. Towards this, Dantchev et al. [26] were able to establish $\Omega(\log n)$ lower bounds on the depth of SP refutations of the Tseitin formulas and the Pigeonhole principle via new techniques which take into account the geometric structure of SP proofs.

6.2 Barriers to Size Lower Bounds

Next, we explore whether it is possible to leverage this depth lower bound in order to obtain size bounds. Throughout this section, we will heavily make use of results of de Rezende, Nordström, and Vinyals [31]. They established a *lifting theorem* that translates *decision tree* lower bounds for a function $f : \{0,1\}^n \to \{0,1\}$ to lower bounds on the real communication complexity of the *composed function* $f \circ \mathsf{IND}_t^n$, which we define next. Let $\mathsf{IND}_t : [t] \times \{0,1\}^t \to \{0,1\}$ be the t-bit *index* function mapping (x, y) to y_x. The function $f \circ \mathsf{IND}_t^n$ is obtained by replacing each variable of f with a copy of IND_t^n on new variables. For any function f, composing with IND_t induces a *standard partition*, where Alice is given $x \in [t]^n$ and Bob is given $y \in \{0,1\}^{tn}$.

The *decision tree complexity* of a function f is closely related to the DPLL complexity of refuting an unsatisfiable formula. A decision tree is a binary tree in which: (i) every internal node is labelled by a variable x_i and has two outgoing edges labelled with 0 and 1, (ii) the leaves are labelled with either 0 or 1. A decision tree computes f if for every $x \in \{0,1\}^n$, the leaf obtained by following the root-to-leaf path which agrees with x is labelled with $f(x)$. The decision tree complexity of f, denoted $\mathsf{DT}(f)$, is the minimal depth of any decision tree computing f.

Theorem 18 (de Rezende et al. [31])**.** *The following statements hold:*

- *For any function $f : \{0,1\}^n \to \{0,1\}$, the real communication complexity of $f \circ \mathsf{IND}_{n^4}^n$ is at least $\mathsf{DT}(f)$.*
- *There is CNF formula F with $\mathsf{poly}(n)$ many clauses which has $\mathsf{poly}(n)$ size resolution refutation but for which any real communication protocol for $\mathsf{Search}_{x,y}(F)$ requires $\Omega(\sqrt{n^{1/4} \log n})$ rounds, for some partition of the variables.*

6.2.1 SP Proofs Cannot be Balanced

As an immediate corollary of Lemma 17 and Theorem 18, we show that SP proofs cannot be balanced. That is, an SP refutation of size s does not imply one of size $\mathsf{poly}(s)$ and depth $\mathsf{poly}(\log s)$. Thus, superpolynomial SP size lower bounds do not immediately follow from depth lower bounds.

Corollary 19. *There exists a CNF formula F which has $\mathsf{poly}(n)$ size SP refutations but any SP refutation requires depth $\Omega(n^{1/8}/\log n)$.*

Proof. This follows immediately by combining Theorem 18 with Lemma 17 together with the fact that SP can simulate resolution proofs. □

6.2.2 Real Communication Cannot be Balanced

Unlike the randomized protocols, the real communication protocols that result from Lemma 17 preserve the topology of the SP proof. That is, the *size* — the number of nodes in the protocol tree — of the resulting real communication protocol is equivalent, up to a $\mathsf{poly}(n)$ factor, to the size of the SP refutation. Therefore, while SP proofs cannot be balanced, one might hope that the resulting real communication protocols could be, and thus size lower bounds could still be obtained from depth bounds. This is not without precedent; both deterministic and randomized communication complexity *can* be balanced. Furthermore, although it known that treeCP cannot be balanced, Impagliazzo et al. [49] show that treeCP refutations of size s can be balanced into $O(\log s)$-round randomized communication protocols for the false clause search problem.

Surprisingly, we show that real communication protocols *cannot* be balanced. To do so, we establish the first lower bound on the real communication of the *set disjointness* function, perhaps the most well-studied function in communication complexity, which we define next. Let $\mathsf{OR}_n : \{0,1\}^n \to \{0,1\}$ be the n-bit $\vee$-function and $\mathsf{AND}_2 : \{0,1\}^n \to \{0,1\}$. Then the set disjointness function, $\mathsf{DISJ}_n := \mathsf{OR}_n \circ \mathsf{AND}_2^n$, is obtained by replacing each of the n input variables of OR_n by a copy of AND_2 on new variables. As before, this function induces a *standard partition* where Alice is given one of the two input bits of each AND_2 function, and Bob is given the other.

Theorem 20. *There is a partition of the variables such that DISJ_n has a real communication protocol of size $O(n)$, but any real communication protocol requires $\Omega((n \log n)^{1/5})$ rounds of communication.*

This lower bound was subsequently improved to $\Omega(n/\log^2 n)$ by Chattopadhyay, Lovett, and Vinyals [16].

The main technique for obtaining lower bounds on the real communication complexity of a function is by a *lifting theorem*, reducing the task of proving lower bounds on certain *composed functions* to the decision tree complexity of the un-composed function. Although DISJ_n is a composed function, there is currently no lifting theorem for composition with the AND_2 function. We circumvent this by exploiting the fact that DISJ_n is *complete* for the class NP^{cc} of functions with polylogarithmic *nondeterministic communication* protocols.

Nondeterministic Communication Complexity. The *nondeterministic communication complexity* of a function $f : \{0,1\}^n \to \{0,1\}$ and a partition (X, Y), is the length of the shortest string $z \in \{0,1\}^\ell$ that can convince Alice and Bob to accept an input $(x, y) \in f^{-1}(1)$ (without communicating). That is, it is the smallest ℓ such that for every $(x, y) \in f^{-1}(1)$ there is a string $z \in \{0,1\}^\ell$ such that both Alice and Bob accept, and for every $(x, y) \in f^{-1}(0)$ and every string $z \in \{0,1\}^\ell$, either Alice or Bob rejects.

To prove the lower bound on DISJ_n we find a function in NP^{cc} to which known lifting theorems for real communication can be applied. Then, we use NP^{cc}-completeness to transfer this lower bound to DISJ_n. The function that we will use is $\mathsf{OR}_n \circ \mathsf{IND}_t$. First, we show that this function belongs to NP^{cc}.

Lemma 21. *There is a $O(\log t + \log n)$ NP^{cc} protocol computing $\mathsf{OR}_n \circ \mathsf{IND}_t^n$ for the standard partition associated with IND_t.*

Proof. Fix some input $(x, y) \in [t]^n \times \{0,1\}^{nt}$ and observe that the ith input bit to OR_n can be computed in $\log t + 1$ rounds of communication by brute-forcing the index gadget: Alice sends $x_i := x_{i,1}, \ldots, x_{i,\log t}$ to Bob who can then compute $\mathsf{IND}_t(x_i, y_i)$, where $y_i := y_{i,1}, \ldots, y_{i,t}$, and return the answer to Bob in a single bit.

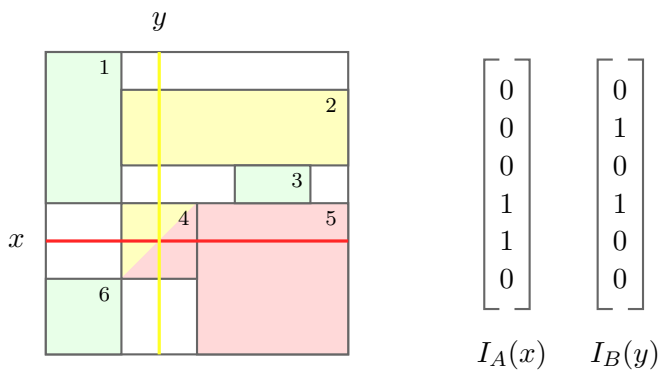

Figure 10: A covering of the communication matrix with monochromatic rectangles (left) and the corresponding DISJ_n instance (right).

Consider the following protocol NP^{cc} for $\mathsf{OR}_n \circ \mathsf{IND}_t^n$: Alice and Bob are given a $\log n$-bit string encoding the index $i \in [n]$ where $\mathsf{IND}_t(x_i, y_i) = 1$; that is, i witnesses that (x, y) is accepting input of $\mathsf{OR}_n \circ \mathsf{IND}_t^n$; see Figure 10. Alice and Bob verify that indeed $\mathsf{IND}_t(x_i, y_i) = 1$ by performing the above brute force protocol in $\log t$ rounds of communication. □

Lemma 22. *Let $m = n^4$, then any real communication protocol computing $\mathsf{OR}_n \circ \mathsf{IND}_t^n$ over the standard partition requires $\Omega(n \log n)$ rounds.*

Proof. Observe that the decision tree complexity of computing OR_n is n (since the *sensitivity* of OR_n is n). The proof follows by combining this with Theorem 18. □

Finally, we are ready to prove the main theorem of this section.

Proof of Theorem 20. First, we prove the lower bound. We will reduce $\mathsf{OR}_n \circ \mathsf{IND}_t^n$ for $t = n^4$ to DISJ_n. By Lemma 21 there is a cover of the 1-entries of the communication matrix of $\mathsf{OR}_n \circ \mathsf{IND}_t^n$ by at most $2nt$ monochromatic rectangles. Enumerating this rectangle covering gives us an instance of set disjointness: on input (x, y) to $\mathsf{OR}_n \circ \mathsf{IND}_t^n$, Alice and Bob construct indicator vectors $I_A(x)$ and $I_B(y)$ of the rectangles in this rectangle covering in their respective inputs lie. Then $\mathsf{OR}_n \circ \mathsf{IND}_t^n = 1$ iff $\mathsf{DISJ}(I_A(x), I_B(y)) = 1$.

This instance of DISJ_n is on $tn/2$ variables, and therefore Lemma 22 implies a lower bound of $\Omega(n \log n)$. Letting $\ell = tn$ be the total number of variables, this is a bound of the form $\ell^{1/5} \log \ell$.

For the upper bound, we give a real communication protocol for $\mathsf{DISJ}_n = \mathsf{OR}_n \circ \mathsf{AND}_2^n$ that has $O(n)$ nodes. Let x, y be the inputs given to Alice and Bob respectively. Sequentially from $i = 1, \ldots, n$, they will solve $x_i \wedge y_i$ by Alice sending x_i to the referee and Bob sending $2 - y_i$. If they discover that $x_i \wedge y_i = 0$ then they halt and output 0, otherwise they continue. □

7 Conclusion

We end with a several questions left open by this work. First, let us note that several of the questions posed in the original version of this paper were subsequently resolved by [25] who exhibited an upper bound (Proposition 4) on the size of the coefficients which occur in Stabbing Planes proofs, and showed that the Tseitin formulas could not provide an exponential separation between Cutting Planes and Stabbing Planes.

1. In this work we showed that there are quasipolynomail-size Stabbing Planes proofs of the Tseitin formulas. Can this be improved to polynomial?

2. A recent work [37] exhibited *supercritical* size/depth tradeoffs for Cutting Planes — exhibiting a formula for which any small proof must have depth which goes far beyond worst-case. This built upon an earlier

supercritical size/width tradeoff for tree-like Resolution by Razborov [9,69]. This is in contrast to sufficiently expressive proof systems, such as AC^0-Frege, which *can* be balanced. Depth captures the degree to proofs — and therefore algorithms which they formalize — can be parallelized. Furthermore, the depth in integer-programming based proof systems such as Stabbing Planes is closely related to rank measures of polytopes, which are studied in integer programming theory. In this work, we showed that Stabbing Planes cannot be balanced. We ask whether this can be improved to a supercritical size/depth tradeoff.

3. We have shown transformations of Cutting Planes proofs into Stabbing Planes which preserve either the size or the depth of the original proof. Does there exist a transformation which preserves both parameters *simultaneously*?

4. Establish super-polynomial lower bounds on the size of Stabbing Planes proofs. As mentioned in the related work section, [35] proved superpolynomial lower bounds on SP proofs with coefficients of magnitude bounded above by 2^{n^δ} for some constant $\delta > 0$ by reducing to Cutting Planes lower bounds, however it is unclear whether this reduction can be made to work for arbitrarily large coefficients. Dadush and Tiwari [25] exhibited an upper bound of $\exp(\mathsf{poly}(n))$ on the magnitude of the coefficients in any Stabbing Planes proof. Thus, one potential (although seemingly unlikely) way to resolve this question is to improve their upper bound to 2^{n^δ}. Another approach for obtaining lower bounds on general Stabbing Planes proofs, suggested by Garg et al. [39], would be to obtain lifting theorem for intersection of triangles trees.

5. Fleming et al. [35] showed that any bounded-weight Stabbing Planes (SP^*) proof can be quasi-polynomially translated into Cutting Planes. Can this simulation be improved to handle SP proofs of arbitrarily large coefficients? Alternatively, can we separate SP from CP?

6. As mentioned in the introduction, we feel that SP has potential, in combination with state-of-the-art algorithms for SAT, for improved performance on certain hard instances, or possibly to solve harder problems such as maxSAT or 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. This opens up the space of algorithmic ideas for solvers and should allow one to take fuller advantage of the expressibility of integer linear inequalities. For example, since geometric properties of the rational hull formed by the set of constraints can be determined efficiently, an SP-based solver could branch on linear inequalities representing some geometric properties of the rational hull. Therefore, it is an open problem to realize a SP based solvers or to implement SP-like branching in conjunction with current solvers.

References

[1] Karen Aardal, Robert E. Bixby, Cor A. J. Hurkens, Arjen K. Lenstra, and Job W. Smeltink. Market split and basis reduction: Towards a solution of the cornuéjols-dawande instances. *INFORMS J. Comput.*, 12(3):192–202, 2000.

[2] Karen Aardal and Arjen K. Lenstra. Hard equality constrained integer knapsacks. *Math. Oper. Res.*, 29(3):724–738, 2004.

[3] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2006.

[4] Albert Atserias, Maria Luisa Bonet, and Juan Luis Esteban. Lower bounds for the weak pigeonhole principle and random formulas beyond resolution. *Inf. Comput.*, 176(2):136–152, 2002.

[5] Egon Balas. An additive algorithm for solving linear programs with zero-one variables. *Operations Research*, 13(4):517–546, 1965.

[6] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical programming*, 58(1):295–324, 1993.

[7] Boaz Barak, Fernando G. S. L. Brandão, Aram Wettroth Harrow, Jonathan A. Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, sum-of-squares proofs, and their applications. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 307–326, 2012.

[8] Amitabh Basu, Michele Conforti, Marco Di Summa, and Hongyi Jiang. Complexity of branch-and-bound and cutting planes in mixed-integer optimization - II. *CoRR*, abs/2011.05474, 2020.

[9] Paul Beame, Chris Beck, and Russell Impagliazzo. Time-space trade-offs in resolution: Superpolynomial lower bounds for superlinear space. *SIAM J. Comput.*, 45(4):1612–1645, 2016.

[10] Paul Beame, Noah Fleming, Russell Impagliazzo, Antonina Kolokolova, Denis Pankratov, Toniann Pitassi, and Robert Robere. Stabbing planes. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 10:1–10:20, 2018.

[11] Eli Ben-Sasson. Hard examples for the bounded depth frege proof system. *Comput. Complex.*, 11(3-4):109–136, 2002.

[12] 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.

[13] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. Lower bounds for cutting planes proofs with small coefficients. *J. Symb. Log.*, 62(3):708–728, 1997.

[14] 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.

[15] Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *J. Comput. Syst. Sci.*, 62(2):267–289, 2001.

[16] Arkadev Chattopadhyay, Shachar Lovett, and Marc Vinyals. Equality alone does not simulate randomness. In Amir Shpilka, editor, *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, volume 137 of *LIPIcs*, pages 14:1–14:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[17] V. Chvátal, W. Cook, and M. Hartmann. On cutting-plane proofs in combinatorial optimization. *Linear Algebra and its Applications*, 114-115:455–499, 1989. Special Issue Dedicated to Alan J. Hoffman.

[18] Vasek Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4):305–337, 1973.

[19] Vašek Chvátal. *Cutting-plane proofs and the stability number of a graph*. Inst. für Ökonometrie und Operations Research, Rhein. Friedrich-Wilhelms-Univ., 1984.

[20] Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, et al. *Integer programming*, volume 271. Springer, 2014.

[21] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979.

[22] William Cook, Ravindran Kannan, and Alexander Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 47(1):155–174, 1990.

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

[24] William J. Cook and Mark Hartmann. On the complexity of branch and cut methods for the traveling salesman problem. In William J. Cook and Paul D. Seymour, editors, *Polyhedral Combinatorics, Proceedings of a DIMACS Workshop, Morristown, New Jersey, USA, June 12-16, 1989*, volume 1 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 75–82. DIMACS/AMS, 1990.

[25] Daniel Dadush and Samarth Tiwari. On the complexity of branching proofs. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 34:1–34:35. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[26] Stefan S. Dantchev, Nicola Galesi, Abdul Ghani, and Barnaby Martin. Depth lower bounds in stabbing planes for combinatorial principles. *CoRR*, abs/2102.07622, 2021.

[27] Sanjeeb Dash. Exponential lower bounds on the lengths of some classes of branch-and-cut proofs. *Math. Oper. Res.*, 30(3):678–700, 2005.

[28] Sanjeeb Dash. On the complexity of cutting plane proofs using split cuts. *Electron. Colloquium Comput. Complex.*, 15(084), 2008.

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

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

[31] Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals. How limited interaction hinders real communication (and what it means for proof and circuit complexity). *Electron. Colloquium Comput. Complex.*, 28:6, 2021.

[32] Santanu S. Dey, Yatharth Dubey, and Marco Molinaro. Lower bounds on the size of general branch-and-bound trees. *CoRR*, abs/2103.09807, 2021.

[33] Yuval Filmus, Pavel Hrubeš, and Massimo Lauria. Semantic versus syntactic cutting planes. In Nicolas Ollinger and Heribert Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, volume 47 of *LIPIcs*, pages 35:1–35:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[34] Matteo Fischetti and Andrea Lodi. Local branching. *Math. Program.*, 98(1-3):23–47, 2003.

[35] Noah Fleming, Mika Göös, Russell Impagliazzo, Toniann Pitassi, Robert Robere, Li-Yang Tan, and Avi Wigderson. On the power and limitations of branch and cut. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 6:1–6:30. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[36] Noah Fleming, Denis Pankratov, Toniann Pitassi, and Robert Robere. Random Θ(log n)-CNFs are hard for cutting planes. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 109–120, 2017.

[37] Noah Fleming, Toniann Pitassi, and Robert Robere. Extremely deep proofs. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 70:1–70:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[38] Nicola Galesi, Dmitry Itsykson, Artur Riazanov, and Anastasia Sofronova. Bounded-depth frege complexity of tseitin formulas for all graphs. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science, MFCS 2019, August 26-30, 2019, Aachen, Germany*, volume 138 of *LIPIcs*, pages 49:1–49:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[39] Ankit Garg, Mika Göös, Pritish Kamath, and Dmitry Sokolov. Monotone circuit lower bounds from resolution. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 902–911. ACM, 2018.

[40] Ralph Gomory. An algorithm for the mixed integer problem. Technical report, RAND CORP SANTA MONICA CA, 1960.

[41] Ralph E Gomory. An algorithm for integer solutions to linear programs. *Recent advances in mathematical programming*, 64(260-302):14, 1963.

[42] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 38:1–38:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[43] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. *SIAM J. Comput.*, 47(5):1778–1806, 2018.

[44] Dima Grigoriev. Tseitin's tautologies and lower bounds for nullstellensatz proofs. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 648–652. IEEE Computer Society, 1998.

[45] Dima Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theor. Comput. Sci.*, 259(1-2):613–622, 2001.

[46] M. Grötschel, L. Lovász, and A. Schrijver. Geometric methods in combinatorial optimization. In William R. Pulleyblank, editor, *Progress in Combinatorial Optimization*, pages 167–183. Academic Press, 1984.

[47] Johan Håstad. On small-depth frege proofs for tseitin for grids. *Electron. Colloquium Comput. Complex.*, 24:142, 2017.

[48] Pavel Hrubeš and Pavel Pudlák. Random formulas, monotone circuits, and interpolation. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 121–131, 2017.

[49] Russell Impagliazzo, Toniann Pitassi, and Alasdair Urquhart. Upper and lower bounds for tree-like cutting planes proofs. In *Proceedings of the Ninth Annual Symposium on Logic in Computer Science (LICS '94), Paris, France, July 4-7, 1994*, pages 220–228. IEEE Computer Society, 1994.

[50] Robert G Jeroslow. Trivial integer programs unsolvable by branch-and-bound. *Mathematical Programming*, 6(1):105–109, 1974.

[51] Hendrik W. Lenstra Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

[52] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In Benjamin Kuipers and Bonnie L. Webber, editors, *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island, USA*, pages 203–208. AAAI Press / The MIT Press, 1997.

[53] Arist Kojevnikov. Improved lower bounds for tree-like resolution over linear inequalities. In João Marques-Silva and Karem A. Sakallah, editors, *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31, 2007, Proceedings*, volume 4501 of *Lecture Notes in Computer Science*, pages 70–79. Springer, 2007.

[54] Jan Krajícek. Discretely ordered modules as a first-order extension of the cutting planes proof system. *J. Symb. Log.*, 63(4):1582–1596, 1998.

[55] Jan Krajek. *Proof complexity*. Cambridge University Press, 2019.

[56] Bala Krishnamoorthy and Gábor Pataki. Column basis reduction and decomposable knapsack problems. *Discret. Optim.*, 6(3):242–270, 2009.

[57] Ailsa H. Land and Alison G. Doig. An automatic method for solving discrete programming problems. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 105–132. Springer, 2010.

[58] László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search problems in the decision tree model. *SIAM J. Discret. Math.*, 8(1):119–132, 1995.

[59] László Lovász and Herbert E. Scarf. The generalized basis reduction algorithm. *Math. Oper. Res.*, 17(3):751–764, 1992.

[60] László Lovász and Alexander Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM J. Optim.*, 1(2):166–190, 1991.

[61] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference, DAC 2001, Las Vegas, NV, USA, June 18-22, 2001*, pages 530–535. ACM, 2001.

[62] Saburo Muroga. *Threshold logic and its applications*. Wiley, 1971.

[63] George L Nemhauser and Laurence A Wolsey. A recursive procedure to generate all cuts for 0–1 mixed integer programs. *Mathematical Programming*, 46(1):379–390, 1990.

[64] Noam Nisan. The communication complexity of threshold gates. *Combinatorics, Paul Erdos is Eighty*, 1:301–315, 1993.

[65] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[66] Toniann Pitassi, Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. Poly-logarithmic Frege depth lower bounds via an expander switching lemma. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 644–657, 2016.

[67] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *J. Symb. Log.*, 62(3):981–998, 1997.

[68] Pavel Pudlák. On the complexity of the propositional calculus. *Sets and Proofs*, 258:197, 1999.

[69] Alexander A. Razborov. A new kind of tradeoffs in propositional proof complexity. *J. ACM*, 63(2):16:1–16:14, 2016.

[70] Grant Schoenebeck. Linear level lasserre lower bounds for certain k-CSPs. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 593–602. IEEE Computer Society, 2008.

[71] Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discret. Math.*, 3(3):411–430, 1990.

[72] João P. Marques Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Trans. Computers*, 48(5):506–521, 1999.

[73] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.