Rank Lower Bounds in Propositional Proof Systems based on Integer Linear Programming Methods

Mark Nicholas Charles Rhodes

A Thesis presented for the degree of
Doctor of Philosophy

Algorithms and Complexity Group (ACiD)
Department of Computer Science
University of Durham
England

March 2009

Dedicated to the lovely Laura.

Rank Lower Bounds in Propositional Proof Systems based on Integer Linear Programming Methods

Mark Nicholas Charles Rhodes

Submitted for the degree of Doctor of Philosophy
February 2009

Abstract

The work of this thesis is in the area of proof complexity, an area which looks to uncover the limitations of proof systems. In this thesis we investigate the rank complexity of tautologies for several of the most important proof systems based on integer linear programming methods. The three main contributions of this thesis are as follows:

Firstly we develop the first rank lower bounds for the proof system based on the Sherali-Adams operator and show that both the Pigeonhole and Least Number Principles require linear rank in this system. We also demonstrate a link between the complexity measures of Sherali-Adams rank and Resolution width.

Secondly we present a novel method for deriving rank lower bounds in the well-studied Cutting Planes proof system. We use this technique to show that the Cutting Plane rank of the Pigeonhole Principle is logarithmic.

Finally we separate the complexity measures of Resolution width and Sherali-Adams rank from the complexity measures of Lovász and Schrijver rank and Cutting Planes rank.

Declaration

The work in this thesis is based on research carried out in the Algorithms and Complexity Group (ACiD) at the Department of Computer Science, Durham University, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

Firstly thanks to all those people who helped me through my studies over the past three or so years and who I will forget to mention in the following paragraphs.

It has been an interesting few years of my life during which I have learnt a lot, met some great people, lived in at least eight different places, got married, played a lot of sport and watched at least one too many DVDs. On the work front the first year or so wasn't too productive and neither was the year I spent trying to solve an annoying problem - the answer to which is 'trivial' but does not appear in this thesis! However the rest of the time was productive enough and for this I must say many thanks to my supervisor Stefan Dantchev who has always been there to answer my questions, however 'trivial' they may have been and who has taught me at least half of all I know and at most one tenth of all he knows about the intriguing field of proof complexity.

Thanks to all the great house-mates I've had during my study period, especially to Paddy and Luke whom I'm sure will go on to lead fun filled lives and who taught me the comedy value of simply repeating the subjective noun of the previous sentence preceded by "You're a"; which sounds a lot more complex and clever than it is.

Thanks also to all my lovely family and my fantastic friends who came to visit us in Durham and Barnard Castle and made our weekends more enjoyable and invariably more drunken.

Thanks also to my good friend Pim van 't Hof, for all the pool and table tennis practice, advice and company, and for teaming up with Laura to cook me enough meals and for lending me enough money to ensure that I didn't waste away.

I would say an extra special thank you to my lovely wife Laura, but she got the dedication instead.

Contents

List of Figures

Chapter 1

Introduction

Some things seem to be easier to prove than others. Indeed some statements appear to have the property that, if true, they have simply defined and "small" proofs, however if the statement turns out to be false, it is much more difficult to consider what constitutes a proof of its invalidity and to see how a "small" proof of this could be produced. For instance consider the statement "there is a book with exactly 100,000 words in it". If this statement is true, then its proof could consist of a book containing exactly 100,000 words and it wouldn't be difficult (although rather time-consuming) to check that the book proves the statement. However, if the statement is not true, then it seems that the only way to prove this would be to count the number of words in every single book that has ever been written.

In fact, some statements may even be more difficult to prove the invalidity of such as, "there is a collection of books whose combined number of words is exactly 10,000,000". It is these kinds of statements which appear to be hard to disprove, no matter which method of attack used or the amount of resources available. The research area in which this thesis is based is Proof Complexity, an area investigating the complexity of proving and disproving such statements. The questions arising from this investigation are related to a number of problems which are fundamental to all areas of mathematics. Given a mathematical theorem, what is the size of the shortest proof of this theorem in some appropriate formal system? Given a formal system, can we say that one tautology is more difficult to prove than another in it? Is it always possible to find a proof which is not much larger than the smallest

possible proof within a reasonable time frame?

The field of modern Proof Complexity was born out of Cook and Reckhow's paper in 1979 [16] which considers the problem of whether or not a given propositional formula is a tautology. One observation in this paper is the realisation that if there exists a proof system which can show every tautology is in fact a tautology in a proof whose size is polynomial in the size of the number of variables in the tautology, then NP = coNP. The reverse statement is also true, namely that if no such proof system exists, then NP $\neq$ coNP, a result which would imply the famous P $\neq$ NP conjecture. The prevailing conjecture is that NP $\neq$ coNP and much work in this area has been dedicated to proving the non-existence of such a proof system, through proving super-polynomial lower bounds for progressively stronger proof systems. Krajíček observes in [47] that although the conjecture NP $\neq$ coNP is unlikely to be settled in this incremental fashion, as a proof of a universal statement is rarely obtained by proving all its instances, it is the hope that such a program will uncover some hidden "computational hardness assumptions" in these lower bounds and thus reduce the conjecture to a more rudimentary one.

The practical application of such lower bounds lies in the fact that if the lower bounds are sufficiently strong, they allow us to show that the tautologies are intractable for any automated theorem proving method (or SAT solver, see [68]) based upon the associated proof system. So far some specific exponential lower bounds are known for a number of systems including general Resolution, and Cutting Planes (see [36] and [52] respectively). Indeed, since most theorem provers are based on "natural" proof systems such as these, another important stream of research in Proof Complexity is devoted to proving the precise limitations of such systems. The work of this thesis is related to this area.

The other main stream of research within proof complexity involves the *automatisability* of proofs for certain proof systems. Work in this area looks to answer the question of whether one can find an algorithm which can always find a proof of size polynomially bounded by the size of the smallest possible proof for a particular proof system. On the positive side little is known; even relatively weak proof systems appear to be non-automatisable (i.e. there is no algorithm which can effectively find

small proofs for those systems). Insights into the automatisability of Resolution and an introduction to the field can be found in [3]. Another question related to the automatisability of proof systems, which we touch on in this thesis, is whether or not a given proof system is automatisable with respect to the smallest possible depth (also known as *rank*) of a proof in that system. We say a proof system is automatisable with respect to the rank if there exists an algorithm which can always find a proof of size $O(n^r)$, where r is the smallest possible rank of a proof of the instance in question.

1.1 Proof Systems and Proofs

Due to the theory of NP-completeness, if one were able to prove a super-polynomial lower bound on the size of proofs of the unsatisfiability of SAT instances, then it would settle the conjecture that NP $\neq$ coNP. It is for this reason, together with the fact that the SAT problem is the most well-studied and understood NP-complete problem, that the field of Proof Complexity deals only at looking at *unsatisfiable* SAT instances, to find lower bounds on the size of proofs of their unsatisfiability.

Perhaps the most intuitive way of checking whether a given propositional formula is satisfiable or not is to simply enumerate over all the 2^n possible values the variables can take and see whether one of them satisfies the formula. This brute-force approach can be represented as a truth table such as in [68], which in turn can be interpreted as a proof generated by a simple proof system. However, this proof system is not particularly interesting to study since trivially all proofs it produces will be of size $\Omega(2^n)$.

A proof system can be considered to be a set of rules for proving that a statement is a tautology (i.e. it is always true). These rules can be considered to be written in the form:

$$\frac{K_1, \ldots, K_q}{D}$$

where the line can read as "from which we can deduce" and q is the number of pieces of information or constraints we already know or have and D is the newly

derived piece of information or constraint. If $q = 0$, D is an axiom of the system (i.e. it is something we need no information to derive). The statements upon which a proof system operates are typically propositional formulae; if this is the case the system is called a propositional proof system (pps). It is well known that given a propositional formula F we can create a CNF formula C from F that is a contradiction (i.e. C is unsatisfiable) if and only if F is a tautology, such that C is only linearly larger than F in linear time (see [67]). Many proof systems, including all those reviewed in this work, operate solely on CNF formulae; such propositional proof systems are also called refutation systems. This name comes from the fact that given an unsatisfiable CNF formula C, one can use the rules of the system to refute the statement "there exists an assignment of true/false values to the variables of C which satisfies C". It should be noted that since the negation of a tautology is a contradiction, the terms tautology and contradiction are used interchangeably throughout the literature.

When discussing a refutation pss, a proof of a CNF formula C can be considered to be an ordered list of statements $L = \{l_1, \ldots, l_s\}$, where each statement l_i is either one of the clauses of C, an axiom of the pps, or is derived from applying one of the system's rules to a collection of statements $l_{i_1}, \ldots l_{i_q}$, where $i_j < i$ for $j = 1, \ldots q$. Here s is usually considered to be the size of the proof (although other definitions of size are more appropriate in certain contexts) and l_s is usually some trivially contradictory statement such as $0 = 1$.

There are two key properties that every useful proof system should have: soundness and completeness. A proof system is said to be *sound* if its rules are logically valid (e.g. in a refutation pps operating on a CNF formula, it should be impossible to derive a contradiction if the original set of clauses are not contradictory). A proof system is said to be *complete* if one can always derive a contradiction when the original set of statements are contradictory by applying its rules. Since we only deal with proof systems with these properties, some obvious questions are therefore: given a proof system P and a tautology T, what is the size of the smallest proof of T in P and how many rounds of applications of the rules of P will suffice to prove T? Propositional proof complexity looks to answer these questions for various

propositional proof systems. For a formal definition of a proof system and further explanation as to the relationship between propositional proof systems and the NP vs coNP problem see [16].

One of the most useful tools for comparing proof systems is the idea of polynomial-simulation (p-simulation). A proof system P is said to p-simulate a proof system P' if for every tautology T, T has a proof of size $O(t^c)$ in P where c is a constant and t is the size of the smallest possible proof of T in P'. If two systems p-simulate each other, they are said to be polynomially equivalent (p-equivalent). A similar tool exists to compare proof systems with respect to proof rank. Since the rank complexity of any formula (i.e. the minimum possible proof rank for any proof of the formula) in a given system generally ranges from 0-n (the number of variables in the formula), we are not interested in comparing the ranks of systems with respect to a polynomial simulation but rather in whether proofs in one system require rank arbitrarily larger than proofs in another system. We say a system P *rank simulates* a system P' if any given tautology has a proof of rank $O(f(n))$ in P, where $f(n)$ is the smallest possible rank of a proof of that tautology in P'. If two systems rank simulate each other they are said to be *rank equivalent*. If however there are examples which require arbitrarily larger rank in one system than in another and vice-versa, the complexity measures of rank in the two systems are said to be *incomparable*.

1.2 Thesis Outline

Chapter 2 begins with basic definitions used throughout the thesis and aims to provide sufficient background knowledge to enable a non-expert to read the thesis. Chapter 2 also provides definitions of the various proof systems of relevance to this thesis together with an overview of results about them. The chapter ends with a figure depicting the current known separations and relationships between the rank complexities of the proof systems discussed together with another complexity measure - Resolution width.

Chapter 3 provides linear rank lower bounds for the Sherali-Adams proof system and also demonstrates that there is a unique connection between the complexity

measures of Sherali-Adams rank and Resolution width. The results presented in this chapter are also presented in the articles [57] and [23].

Chapter 4 presents a novel method to proving rank lower bounds for the Cutting Planes proof system. This method is used to show that the rank of the well-known Pigeonhole Principle polytope is $\Theta(\log n)$, where n is the number of "holes". The chapter also provides a short proof of the fact that the principle has polynomially sized proofs, showing that Rank lower bounds do not necessarily imply size lower bounds in the Cutting Planes proof system. The results from Chapter 4 are contained in the manuscript [58].

In Chapter 5 we demonstrate that the complexity measures of Cutting Plane rank and the rank of the proof systems proposed by Lovász and Schrijver are incomparable to the complexity measure of Resolution width. This result together with a result from Chapter 3 provides a number of separations between the various proof systems discussed in this thesis. Chapter 5 draws from results appearing in [59] and [60].

Chapter 2

Preliminaries

2.1 Background

2.1.1 Computation and Complexity

We assume the reader is familiar with the basic concepts of the theories of computation and complexity such as problems, algorithms and asymptotic bounds. For excellent introductions to computation and complexity theory see [64] and [20] respectively. We begin by defining the complexity classes discussed in this thesis.

P is the set of problems solvable by a deterministic Turing machine in polynomial time in the size of the input.

NP is the set of problems that have short certificates, i.e. a given solution to a problem in this set can be verified as a solution in polynomial time in the size of the input. Equivalently NP is the set of problems that can be solved by a non-deterministic Turing machine in polynomial time.

coNP is the set of problems that have short disqualifications, i.e. given a potential proof that an instance of a problem in coNP can not be solved, it can be verified in polynomial time that this is indeed a proof.

A problem is NP-complete (coNP-complete) if it is in NP (coNP) and is at least as hard as any other problem in NP (coNP). By the term "at least as hard" we mean that every other problem in NP (coNP) can be reduced to that problem in polynomial time. This definition leads to an interesting fact; if any problem known

to be NP-complete (coNP-complete), were shown to be solveable in polynomial time, it would mean that every problem in NP (coNP) could also be solved in polynomial time. It is widely believed that no polynomial time algorithm exists for any NP-complete or coNP-complete problem. The problem of whether or not there exists a polynomial time algorithm for any NP-complete problem is famously known as the P vs NP problem and its solution is worth a million dollars [17]. Since many real-world problems are known to belong to these classes, the development of a such an algorithm would be an unrivalled breakthrough. For a thorough treatment of the theory of NP-completeness see [32].

2.1.2 Graphs

Throughout this thesis we use some basic concepts from the theory of graphs, which we will define here. For a more thorough introduction to the theory of graphs see [10].

An undirected graph $G = (V(G), E(G))$ is a set of *vertices* (also known as nodes) $V(G)$ and a set of *edges* (also known as arcs) $E(G)$. For simplicity, whenever it is unambiguous we refer to $V(G)$ as simply V and $E(G)$ as E. Each edge $e \in E$ is an unordered pair uv where $u, v \in V$. We say that e *connects* u and v. We only deal with simple graphs which have no *loops* (edges of the form vv for some vertex $v \in V$) and no *multiple edges* (i.e. all edges in E are distinct). Graphs are often represented as a diagram, where the vertices are indicated by points and edges by lines joining the points representing the vertices it connects. We say two vertices are *adjacent* if and only if they are connected by an edge, and we say e is *incident* to v if it connects v to some other vertex. The *neighborhood* of a vertex v, denoted as $\Gamma(v)$, is the set of all vertices that v is adjacent to. In a simple graph, the size of $\Gamma(v)$ is equal to the number of edges that are incident to v; this number is called the *degree* of v.

A *connected graph* is a graph in which each vertex can be reached from every other vertex by traversing edges of the graph. A *clique* refers to a set of vertices where each pair of vertices are connected by an edge. In contrast, an *independent set* is a set of vertices that have no edges between them. A *subgraph* H of G is a

graph consisting of a subset of the edges and a subset of the vertices of G (i.e. a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$). A *cycle* in a graph is a sequence of unique vertices and edges (also known as a path) that can be traversed starting from a vertex v so that you end up back at v. An interesting class of graphs, relevant to Proof Complexity, are those which are connected and contain no cycles; such graphs are called *trees*.

Directed graphs are defined similarly to undirected graphs with the exception that the edges in a directed graph are *ordered* pairs of vertices (i.e. the edge uv is different from the edge vu); we say that the edge uv leaves u and goes to v and is typically represented as an arrow from u to v in a drawing of the graph. For such graphs the *out-degree* of a vertex is thenumber of edges leaving it, whereas the *in-degree* is the number of edges going to it. One type of directed graph commonly used in Proof Complexity to represent proofs are *Directed Acyclic Graphs* (or DAGs for short). As the name suggests a DAG is simply a directed graph with no cycles. A *directed tree* is a DAG with the additional condition that the in-degree of any vertex is at most one. A *root* in a DAG is any vertex with in-degree zero, whereas a *sink* is a vertex with out-degree zero. In a DAG, a vertex u is called a *parent* of a vertex v if the edge uv appears in the DAG; likewise v is a *child* of u in this instance.

2.1.3 Propositional logic and the Satisfiability Problem

A propositional formula is a sentence constructed from propositional variables, a set of connectives and parentheses (brackets). A propositional variable can take only one of two possible values, *true* or *false*, which are often denoted as the numbers 1 and 0 respectively. Each variable can be considered to represent a particular statement; for example, the propositional variable x might represent the statement "Tom is a cat" and the variable y the statement "Tom likes milk". The basic connectives are conjunction, disjunction and negation. Conjunction also known as "and", is represented with the symbol $\wedge$; disjunction also known as "or" is represented by the symbol $\vee$; and negation also called "not" is denoted by $\neg$. Often other connectives such as implication ($\rightarrow$) and exclusive-or ($\oplus$) are also used; although these do not add to the expressive power of the language, they can be used to simplify some

formulae. The semantics of the basic connectives is as follows:

- $\neg p$ means that p is false. Also $\neg\neg p$ is the same as p.
- $p \vee q$ denotes that at least one of p and q must be true.
- $p \wedge q$ says that both p and q must be true.

Note that in the above, p and q can be propositional variables or entire subformulas and that negation has a higher precedence than the other two connectives.

An example of a propositional formula, using the variables x and y introduced earlier, is $F = (\neg x \wedge \neg y) \vee (x \wedge y)$, which states that "Tom isn't a cat and doesn't like milk or Tom is a cat who likes milk". The literals of a formula are its variables or their negations (i.e. the literals of F are x, $\neg x$, y and $\neg y$).

Any given assignment of true/false values to the variables of a propositional formula gives a true/false value for that formula. This is obtained simply by filling in the values into the formula and applying the rules of the connectives. An assignment of values to the variables of a formula which makes the formula evaluate to *true* is known as a *satisfying assignment*. For a formula with l literals and c connectives, given an assignment of values to variables, we can find the 0/1 value of the formula in time $O(l+c)$. A formula with at least one satisfying assignment is called *satisfiable*, one which is satisfied by every assignment is known as a *tautology*, whilst a formula which has no satisfying assignments is called *contradictory*. There are 2^n possible 0/1 values the variables can take for any formula containing n variables. A table which gives the 0/1 value of a formula for each possible assignment of values to its variables is called a truth table. The truth table for the basic connectives and F is is given in Table 2.1.

The question of whether there exists an assignment of values to the propositional variables of a formula so that the formula evaluates to true is known as the satisfiability problem or SAT for short. For instance F has two satisfying assignments, and is therefore neither a tautology nor a contradiction.

x	y	$\neg x$	$x \wedge y$	$x \vee y$	F
0	0	1	0	0	1
0	1	1	0	1	0
1	0	0	0	1	0
1	1	0	1	1	1

Table 2.1: Truth table for the basic connectives and the formula F.

SAT is typically studied for formulae in conjunctive normal form (CNF). A formula is in CNF if it consists of a conjunction of disjunctions of literals. Each set of disjunctions is called a clause, and we write:

$$F := C_1 \wedge \ldots \wedge C_m = \bigwedge_{k=1}^{m} C_k,$$

and each clause C_k is of the form:

$$\bigvee_{i \in P_k \cup N_k} l_i \equiv \bigvee_{i \in P_k} p_i \vee \bigvee_{i \in N_k} \neg p_i,$$

where l_i is a literal and p_i are the propositional variables. Here N_k (P_k) are the indices of the negative (positive) literals of the clause C_k. The length of a clause C_k is the number of distinct literals in it. We consider tautological clauses (i.e. those with $P_k \cap N_k \neq \emptyset$) to have length zero since they evaluate to *true*. We call a clause a k-clause if it has length k, and a formula is said to be a kCNF if it contains only clauses of length $\leq k$. The satisfiability problem restricted to kCNF formulae is known as kSAT. The rational behind restricting the study of SAT to CNF formulas comes from the fact that for any propositional formula we can generate a CNF formula which is satisfiable if and only if the original formula is satisfiable in linear time, (with only linearly more variables). For details on how this can be achieved we refer the reader to [67].

SAT is known to be NP-complete and in fact was the first problem to be shown to be so by Cook in his ground breaking paper [15]. In fact, even when restricted to 3CNF formulae the problem remains NP-complete, (see Theorem 34.10 of [20]).

The SAT problem is also very important from a practical standpoint. It has a large number of applications and its simplicity makes it easy to convert instances of

many other NP-complete problems into SAT instances. Advances in algorithms to solve SAT, so-called SAT solvers, over the past two decades has lead to real-world instances consisting of several thousands of variables being solved (see [68]). Currently SAT solvers are used to solve problems in planning, machine learning, digital integrated circuit design, frequency assignment, test pattern generation, haplotyping in bioinformatics and model checking amongst others; for further information, see [50].

2.1.4 Polyhedra, Linear Inequalities, Linear Programming and Integer Linear Programming

A set $P \subseteq R^n$ is a *polyhedron* if $P = \{x \in \mathbb{R} : Ax \geq b\}$ for some matrix A and some vector b, and a *rational* polyhedron if A and b can be taken to be rational. We say that P is *described* or *defined* by the system $Ax \geq b$. A *polytope* is the name given to a bounded polyhedron (i.e. all its variables have finite maximum and minimum values). Every polyhedron P has an *integer hull* (i.e. the convex hull of the integer points within P), which we denote as P_I and which may or may not be empty.

The Linear Programming problem (LP) is the problem of optimizing a linear objective function, subject to a set of (also called a *system* of) linear equality and inequality constraints. LP can be stated as

$$\begin{aligned} &\text{maximize} && c^T x \\ &\text{subject to} && a_i^T x \geq b_i,\ i = 1, \ldots, m \end{aligned}$$

where $a_1, \ldots, a_m$ and c are rational vectors of n elements and $b_1, \ldots, b_m$ are rational numbers. This problem has numerous real-world applications and has been the subject of much research over the last sixty years. The first and probably the best known algorithm to solve the linear programming problem is the Simplex method. This method was developed by Dantzig, who began the modern study of the problem when developing a method to rapidly compute a time-staged deployment, training and logistical supply program for the Pentagon. For a brief history of history of LP see [24]. However, whilst it was observed that LP could be solved quickly in practice, the complexity of LP was not known until Khachiyan [44] demonstrated in 1979 that

the ellipsoid method could be used to solve LP in polynomial time for all bounded polyhedra. In this work we are only concerned with bounded polyhedra in the 0-1 hypercube (i.e. those of the form $P \subseteq [0,1]^n$) and with the decision version of the problem, which asks whether there exists any point which satisfies the constraints $a_i^T x \geq b_i$, $i = 1, \ldots, m$. Obviously, since the optimization version of the problem is known to be in P, so too is the decision version.

One important related problem is Integer Linear Programming (ILP). This is essentially the same problem, but, as its name suggests, has the additional restriction that each variable must take an integer value. ILP is well known to be NP-hard. In fact the decision problem version (i.e. deciding whether the given constraints can be satisfied at all or not) in which all the variables must be either 0 or 1, (the 0/1 Integer Programming Problem) was one of the first problems to be shown to be NP-complete, by Karp [43]. Both ILP and its continuous counter-part have been thoroughly studied, for an excellent overview of this study see [62].

2.2 Resolution

Resolution is by far the most studied and understood proof system which doesn't simply enumerate over all possible inputs. It was first introduced as a proof system by Blake in [9] and revisited some thirty years later by Davis and Putnam in [27] and [28] when it began to be used as a method for solving the SAT problem. Resolution has only one rule, known as the Resolution rule:

$$\frac{A \vee x_i \quad B \vee \neg x_i}{A \vee B}$$

where A and B are disjunctions of literals.

In the case that the length of A or B is 0, this rule is sometimes referred to as *unit propagation*. In addition to the Resolution rule, the Weakening rule is sometimes added as a means of simplifying arguing about Resolution proofs, however its inclusion adds no additional power to the system. The Weakening rule is defined as:

$$\frac{A}{A \vee l}.$$

Here l is a literal, which is either a positive or negative occurrence of a variable not already in A.

The clause resulting from an application of the Resolution rule is sometimes referred to as the *resolvent* of the two given clauses and the variable which is removed when creating the resolvent is said to be *resolved* upon. A proof of the unsatisfiability of a CNF formula F in Resolution (also called a Resolution proof or *refutation*), is simply a list of clauses $C_1, \ldots, C_s$ where each C_i is either a clause of F or is the resolvent of $C_{i'}$ and $C_{i''}$ where $i > \{i', i''\}$ and C_s is the empty clause ($\emptyset$). The *size* of a Resolution proof is defined to be the the number of clauses in it. As with other proof systems, Resolution refutations can be represented as a DAG, where the vertices of the graph are the clauses appearing in the proof and there is an edge in the graph if and only if the target was derived using the source as a premise. Each edge can be labeled with the variable which was resolved upon to create the target. The clauses of the formula are the only sources of the graph (vertices with in-degree 0) and the empty clause ($\emptyset$) the only sink. Figure 2.1 shows a simple example of a Resolution refutation of the formula $(x \vee y) \wedge (\neg y \vee z) \wedge (\neg x \vee y) \wedge \neg z$ represented in this manner; such graphs are sometimes called refutation graphs.

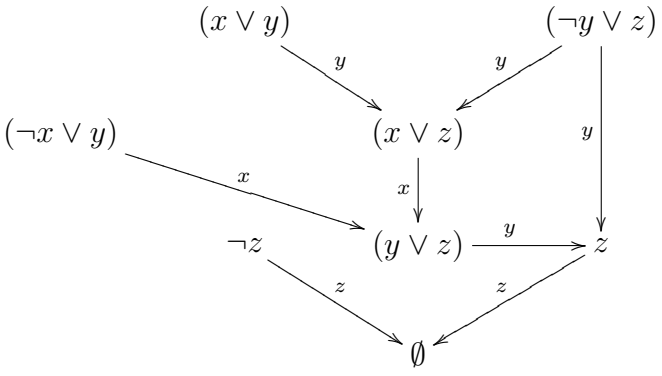


Figure 2.1: A simple refutation graph.

It is well known that Resolution is both sound (since the resolvent of any C_i and $C_{i'}$ can only be satisfied if both C_i and $C_{i'}$ can be satisfied), and complete by Theorem 2.2.1, the proof of which we include for completeness and because it is hopefully illuminating.

Theorem 2.2.1 (Adapted from Theorem 2.1.1 of [47]) If a CNF formula F is contradictory, then there is a Resolution proof of F.

Proof: We use induction on the number of variables n in F. In the case that $n = 1$ it must be that we have the two clauses x_1 and $\neg x_1$, which resolve to give the empty clause.

Assume there is a Resolution proof for any contradictory CNF formula with $n-1$ variables. We can partition the clauses of $F = \{C_1, \ldots, C_m\}$ into four disjoint sets; $C_{00}, C_{01}, C_{10}, C_{11}$; these sets contain all the clauses of F without the literals x_n or $\neg x_n$, those with the literal x_n but without $\neg x_n$, those without x_n but with $\neg x_n$ and those with both x_n and $\neg x_n$ respectively.

We now create a new set of clauses F' from F as follows:

1) Delete all clauses from C_{11}, since they are trivially satisfied.

2) Replace $C_{01} \cup C_{10}$ by the set of clauses that are obtained by resolving all pairs of clauses $C_i \vee x_n \in C_{01}$ and $C_j \vee \neg x_n$ from C_{10}.

Note that F' contains no occurrence of the literals x_n or $\neg x_n$. Moreover F' can also be shown to be unsatisfiable. This is because any satisfying assignment a' of zero-one values to the variables of F' satisfies at least all the clauses C_i, such that $C_i \vee x_n \in C_{01}$, or all the clauses C_j, such that $C_j \vee \neg x_n \in C_{10}$, (otherwise we could find some resolvent in F' not satisfied by a'). However if this were the case a' could be extended, by giving a suitable value for x_n to a truth assignment a satisfying F, which is a contradiction. □

There are two well studied restrictions of Resolution: Regular Resolution and Tree-Like Resolution. In the former each variable is resolved at most once along any path in the refutation graph; in the latter restriction, the refutation graph is a tree. Both are known to be exponentially weaker than general (unrestricted) Resolution (i.e. there exist examples of families of contradictions that are known to have polynomially sized general Resolution proofs and require at least exponentially sized Regular Resolution and Tree-Like Resolution proofs).

An interesting complexity measure unique to Resolution is the *width* of a Resolution proof. The measure of the width of a Resolution refutation was introduced in [7] and is defined to be the size of the largest clause in the refutation. The Resolution width of a CNF formula F is the minimum possible width of any Resolution refutation of F. It is known that the width and the size of Resolution and in particular Tree-like Resolution proofs are directly correlated under certain conditions (see [7]).

An alternative definition of the Resolution width of a CNF formula F, known as the *narrow* Resolution width, comes from [31]. This refined definition of width overcomes the problem that instances with large clauses must have high width, even though it may be possible to reduce the width simply by rewriting the clauses as a 3CNF formula. Details on how this can be accomplished, using auxiliary variables, are given in Theorem 34.10 of [20]. The narrow Resolution width of a CNF formula F is the minimum value k, such that F has a *width k narrow Resolution* refutation. A width k narrow Resolution refutation is a sequence of clauses $C_1, \ldots, C_s$ where C_s is the empty clause and each C_i is either a clause in F or is derived using either the Resolution rule, the Weakening rule or the following rule, which is referred to as Resolution by cases

$$\frac{x_1 \vee \cdots \vee x_m, \quad B \vee \neg x_1, \ldots, B \vee \neg x_m}{\mathrm{B}}.$$

Crucially all clauses derived from any of the three rules must have at most k literals. In [31], it is shown how the narrow Resolution proof system is related to normal Resolution.

Tree-like Resolution, a relatively weak proof system, is a particularly interesting system to study from a practical point of view, because many of the most effective SAT solvers, including the famous DPLL (Davis, Putnam, Loveland and Longmann [27, 28]) algorithm, are based upon it. It may seem counter-intuitive that practical algorithms are based on one of the weakest proof systems, but this can be explained by the trade-off between strength and automatisability, i.e that the weaker the proof system, the more likely it is to be automatisable.

2.3 Proof Systems Operating on Linear Inequalities

In general there are two types of propositional proof system (pps), those such as Resolution which are based upon solving the SAT problem directly and those based upon solving instances of ILP derived from given SAT instances. A SAT instance can be converted into an ILP instance by taking each clause of the instance and replacing it with an inequality as follows:

the clause

$$x_{i_1} \vee \cdots \vee x_{i_t} \vee \neg x_{j_1} \vee \cdots \vee \neg x_{j_f}$$

becomes the inequality

$$x_{i_1} + \cdots + x_{i_t} + (1 - x_{j_1}) + \cdots + (1 - x_{j_f}) \geq 1.$$

We then add the restriction that all the variables must take either the value zero or one, this can be expressed as the set of quadratic equalities ${x_i}^2 - 1 = 0$ for all variables x_i. Notice that the solutions to this ILP are exactly the satisfying assignments of the original CNF formula.

The ILP methods we consider all operate in a similar manner. They firstly remove the restriction that each variable must take an integer value. In its place, the bounding inequalities $0 \leq x_i \leq 1$ are added to the initial set of inequalities for each variable x_i. The result is a system of linear inequalities which may have many fractional solutions, but, not necessarily any integer solutions. However, since all the constraints are linear, if a fractional solution exists it can be found in polynomial time. Notice that if the orignal ILP instance is derived from a set of CNF clauses as above, then setting all the variables to exactly $\frac{1}{2}$ in the resulting system of linear inequalities will satisfy every inequality derived from clauses with at least two variables. Polytopes defined with these bounding inequalities are sometimes referred to as being in the 0-1 cube, or equivalently, the 0-1 n-dimensional hypercube. The ILP methods then derive more new inequalities, which are sometimes referred to as *cuts*, and add them to this system until eventually they reach its integer hull; the ILP instance is solvable if and only this is non-empty.

The idea of using ILP methods as propositional proof systems through this conversion was first proposed in [18].

2.3.1 Cutting Planes

The Cutting Planes (CP) method was first introduced as a means of solving ILP in [34] and was first considered as a proof system in [18]. This method is sometimes referred to as the Gomory-Chvátal cutting planes method in the literature. The CP proof system, can be considered as a refutation proof system (i.e. it derives the contradiction $1 \leq 0$) operating on a system of linear inequalities, generated by converting a CNF as described above, which has just one inference rule, namely the cut rule:

$$\frac{\begin{array}{c} a_{11}x_1 + \cdots + a_{1n}x_n \geq b_1 \\ \cdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n \geq b_n \end{array}}{(\sum_{i=1}^{m} \lambda_i a_{i1})x_1 + \cdots + (\sum_{i=1}^{m} \lambda_i a_{in})x_n \geq \lceil \sum_{i=1}^{m} \lambda_i b_i \rceil}$$

where $m \leq n$ and the λ_i's are non-negative real coefficients satisfying $\sum_{i=1}^{m} \lambda_i a_{ij} \in \mathbb{Z}$ for all $1 \leq j \leq n$.

CP proofs can be represeted as DAGs in which the inequalites are shown as vertices and all non-root vertices are derived by applying the cut rule to the inequalities that have an edge to them. The edges can be labelled by the coefficient by which the source was multipled when deriving the target. Figure 2.2 shows the DAG representing a simple CP proof of the unsatisfiability of the system of inequalities derived from the CNF formula $(x \vee y) \wedge (\neg y \vee z) \wedge (\neg x \vee y) \wedge \neg z$ as in Section 2.3. In this figure,

The rank of a CP proof is the number of edges in the largest path in the DAG representing it. The size of the proof is the number of vertices in the DAG. For example, the rank of the proof in Figure 2.2 is two and the size of the proof is six. The CP rank of an inconsistant system of inequalities is defined to be 0; the CP

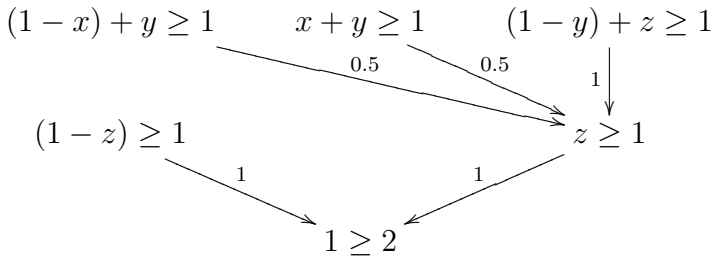


Figure 2.2: A simple CP proof.

rank of a consistant system of inequalities is defined as the minimum possible rank of a CP proof for them. This measure is sometimes referred to as the Chvátal Rank of the inequalities and has been well studied (see [37]). When studying the CP rank it is often useful to consider the following definition of the polytope that can not be removed by applying the cut rule to a given polytope, as presented in [12]. In the definition P refers to the current polytope, $\langle\rangle$ to the standard dot or inner product and P' to the polytope remaining after a round of applications of the cut rule:

$$P' = \{x \in P : \langle ax \rangle \geq \lceil b \rceil \text{ whenever } a \in \mathbb{Z}, b \in \mathbb{R}, \text{ and } \langle ay \rangle \geq b \text{ for all } y \in P\}.$$

It follows from the fact that CP can simulate Resolution, such that the simulation is rank preserving (i.e. one can convert a Resolution proof of size s and rank r into a CP proof of rank r and size $O(s^c)$ where c is a constant, as shown in [18]) that the rank of any polytope corresponding to a set of unsatisfiable clauses is at most n.

2.3.2 Lovász Schrijver

There are three distinct proof systems studied in proof complexity which were devised by Lovász and Schrijver in [49]; from the weakest to the strongest they are LS_0, LS, LS_+. These systems can be defined with respect to the rules with which they can derive new inequalities.

Of relevance to LS_+ is the concept of a *positive semidefinite* (PSD) matrix. A matrix A is said to be PSD if $x^T Ax \geq 0$ for all vectors x. Equivalently, if A is a square symmetric matrix (i.e. A is an $n \times n$ matrix for some natural number $n \geq 1$ and $A_{(i,j)} = A_{(j,i)}$ for all i and j) and $A = U^T U$ for some other matrix U, then

A is positive semidefinite [39]. Another fact about these matrices we use is that if $A = B + C$ and B and C are both PSD, then so is A.

Given a polytope $P \subseteq [0,1]^n$ defined by the set of inequalities $\langle a_i x \rangle \geq b_i$, where $1 \leq i \leq m$, the inequality $\langle cx \rangle \geq d$ is called an N-cut for P if

$$\langle cx \rangle - d = \sum_{i,j} \alpha_{i,j}(\langle a_i x \rangle - b)x_j + \sum_{i,j} \beta_{i,j}(\langle a_i x \rangle - b)(1 - x_j) + \sum_j \lambda_j(x_j^2 - x_j),$$

where $\alpha_{i,j}, \beta_{i,j} \geq 0$, $\lambda_j \in \mathbb{R}$ for all $1 \leq i \leq m$, $1 \leq j \leq n$. Note that this is a linear combination of the inequalities multiplied by positive and negative literals and that the role of the term involving the λ_j is to remove any occurrences x_j^2 and replace them with x_j.

An N_0-cut is defined similarly as an N-cut except that when projecting back the linear term, $x_i x_j$ is viewed as distinct from $x_j x_i$.

The inequality $\langle cx \rangle \geq d$ is an N_+-cut if

$$\begin{aligned}\langle cx \rangle - d = &\sum_{i,j} \alpha_{i,j}(\langle a_i x \rangle - b)x_j + \sum_{i,j} \beta_{i,j}(\langle a_i x \rangle - b)(1 - x_j) \\ &+ \sum_j \lambda_j(x_j^2 - x_j) + \sum_k (\langle h_k x \rangle + g_k)^2,\end{aligned}$$

where $\alpha_{i,j}, \beta_{i,j}$ and λ_j are defined as in an N-cut and $\langle h_k x \rangle + g_k$ is simply any linear expression for $k = 1, \ldots, n+1$, not necessarily related to the set of defining inequalities.

This is a valid inference since the square of any linear expression is clearly positive. An N_+-cut is often called a semidefinite cut which is due to the fact that the factor $\sum_k (\langle h_k x \rangle + g_k)^2$ can be described as a PSD matrix.

As with other proof systems we can view proofs in the Lovász and Schrijver systems as an ordered set of inequalities $c_1, \ldots c_s$, where c_s is a trivially inconsistant inequality such as $0 \geq 1$, and each c_i is either one of the initial inequalities or is some type of cut created from a set of inequalities, each of which appears before c_i in the list. In the system LS_0, only N_0-cuts are allowed, LS allows N-cuts and N_+-cuts are allowed in LS_+. In this definition s is considered to be the size of the proof; the rank of the proof is the length of the longest path in the DAG representing the proof

(i.e. the DAG in which the inequalities of the proof are the vertices and there is an edge if and only if the target was derived from the source). As with CP, the size of system of inequalities is the minimum possible size of a proof of them in the given system. In all the LS systems the rank of an inconsistant system of inequalities is zero, whilst the rank of a consistant system of inequalities is the minimum possible rank of any proof of them. It is known that any polytope derived from a set of unsatisfiable clauses has LS_0 rank of at most n, which in turn implies that the same holds for the two stronger systems.

When working with the Lovász-Schrijver systems it is often useful to consider alternative definitions of them, first presented in [25], allowing one to clearly identify points within the polytope which survive a round of cuts. These alternative definitions are most easily stated for the projective cone $\bar{P} \in \mathbb{R}^{n+1}$ of a polytope $P \subseteq [0,1]^n$, where we have that

$$\bar{P} \equiv \{(a, aw_1, \ldots, aw_n) : a \geq 0 \text{ and } (w_1, \ldots w_n) \in P\}.$$

Since a point in P is given by a set of values for $x_1, \ldots, x_n$, we refer to the extra coordinate as x_0. It should be noted that P is exactly the intersection of $\bar{P}$ with the hyperplane $x_0 = 1$, if the point $w = (w_0, w_1, \ldots, w_n)$ is in $\bar{P}$, it means that the point $(w_1/w_0, \ldots, w_n/w_0) \in P$. The exception is when $w_0 = 0$; all points having $x_0 = 0$ are defined as being in P, so long as $P \neq \emptyset$. Let Y be an $(n+1) \times (n+1)$ matrix and let e_i be the vector of length $n+1$ with all elements set to zero except element i which is one. In the following definition $\bar{P}'$ is the polytope defined by all inequalities that can be derived from the set of inequalities defining $\bar{P}$ and a single application of the respective cut rule.

(i) A point $w \in \mathbb{R}^{n+1}$ is in $\bar{P}'$ for LS_0 if there is an $(n+1) \times (n+1)$ matrix Y such that $Ye_0 = (e_0^T Y)^T = diag(Y) = w$ and for all i, $Ye_i \in \bar{P}$ and $Ye_0 - Ye_i \in \bar{P}$.
(ii) A point $w \in \mathbb{R}^{n+1}$ is in $\bar{P}'$ for LS if (i) holds and the matrix Y is symmetric.
(iii) A point $w \in \mathbb{R}^{n+1}$ is in $\bar{P}'$ for LS_+ if (ii) holds and the matrix Y is also positive semidefinite.

To see that this definition is equivalent to the initial definition, in the sense that we have that $P' = \bar{P}' \cap [x_0 = 1]$ being the polytope remaining after applying a single round of applications of the respective cut rule, we refer the reader to [25].

2.3.3 Sherali-Adams

The Sherali-Adams (SA) operator was introduced in [63] as a method for generating a hierarchy of relaxations for linear and polynomial 0-1 programming problems. In this review we discuss the operator as a proof system. The SA operator is sometimes referred to as a *static* proof system. This means that it simply generates a set of linear inequalities which can either be proven to have integer solutions or not, with the use of a polynomial time LP algorithm. There is no need to consider the computational path taken to obtain the contradiction or solution.

When using the SA proof system we label the variables with sets P and N and write $\chi_{[P|N]}$ to mean the variable labelled with P and N. We also say that the variable $\chi_{[P|N]}$ has *positive side* P and *negative side* N. To improve the clarity of the presentation we drop the brackets around sets containing a single element (i.e. we would denote $\chi_{[P\cup\{i\}|N]}$ as $\chi_{[P\cup i|N]}$). Intuitively, the variables used by the SA proof system represent partial assignments to the propositional variables of the original CNF formula. The "positive" ("negative") side of an SA variable contains the propositional variables that are set positively (negatively) in the partial assignment the SA variable represents.

To use the SA operator on a given ILP instance Φ, we initially rewrite Φ by replacing each occurrence of the variable x_i in each inequality $\varphi \in \Phi$ with the variable $\chi_{[x_i|\emptyset]}$. Let $k \geq 0$ be an integer. The Sherali-Adams formulation of rank k (also referred to as the k-th lift or the level-k formulation) derived from a set of converted inequalities Φ is the following:

It has a variable $\chi_{[P|N]}$ for every pair of disjoint subsets P and N of all variables appearing in Φ, where $|P \cup N| \leq \min\{k+1, n\}$. We add the inequalities $\chi_{[P|N]} \leq 1$ and $\chi_{[P|N]} \geq 0$ for each variable $\chi_{[P|N]}$ in the system, since these bound the range of

each variable, they are sometimes referred to as the *bounding inequalities*. We also add the following constraint:

$$\chi_{[\emptyset|\emptyset]} = 1.$$

For all variables $\chi_{[P|N]} \in \Phi$ and all variables i in the original instance, where $|P \cup N| \leq k$ and $i \notin P \cup N$, we add:

$$\chi_{[P \cup i|N]} + \chi_{[P|N \cup i]} = \chi_{[P|N]}.$$

These equalities are sometimes referred to as the *equalities of negation*. They also ensure that $\chi_{[P \cup i|N]}, \chi_{[P|N \cup i]} \leq \chi_{[P|N]}$; this property is sometimes referred to as *monotonicity*.

Each inequality $\varphi_i \in \Phi$ *multiplied* by each variable $\chi_{[P|N]}$, where $|P \cup N| \leq k$. The variable $\chi_{[A|B]} \in \varphi_i$, multiplied by $\chi_{[P|N]}$, becomes $\chi_{[A \cup P|B \cup N]}$. If $(A \cup P) \cap (B \cup N) \neq \emptyset$ the variable is assigned the value 0. We refer to the deriving of a new inequality by multiplying by a variable $\chi_{[P|N]}$ where $|P \cup N| = 1$ in this way as an SA multiplication.

As an example, suppose we begin with the system of inequalities derived from the CNF formula $(x \vee y) \wedge (x \vee \neg y)$, i.e. $x + y \geq 1$, $x \geq y$, $0 \leq x \leq 1$ and $0 \leq y \leq 1$. The rank one SA formulation of this is the combination of the following:

- The initial set of inequalities rewritten:
  $\chi_{[x|\emptyset]} + \chi_{[y|\emptyset]} \geq 1$, $\chi_{[x|\emptyset]} \geq \chi_{[y|\emptyset]}$, $0 \leq \chi_{[x|\emptyset]} \leq 1$ and $0 \leq \chi_{[y|\emptyset]} \leq 1$.
- Further bounding inequalities:
  $0 \leq \chi_{[\emptyset|x]} \leq 1$, $0 \leq \chi_{[\emptyset|y]} \leq 1$, $0 \leq \chi_{[\{x,y\}|\emptyset]} \leq 1$, $0 \leq \chi_{[x|y]} \leq 1$, $0 \leq \chi_{[y|x]} \leq 1$ and $0 \leq \chi_{[\emptyset|\{x,y\}]} \leq 1$.
- The equalties of negation:
  $\chi_{[x|\emptyset]} + \chi_{[\emptyset|x]} = \chi_{[\emptyset|\emptyset]}$, $\chi_{[y|\emptyset]} + \chi_{[\emptyset|y]} = \chi_{[\emptyset|\emptyset]}$, $\chi_{[x|y]} + \chi_{[\{x,y\}|\emptyset]} = \chi_{[x|\emptyset]}$, $\chi_{[y|x]} + \chi_{[\emptyset|\{x,y\}]} = \chi_{[\emptyset|x]}$, $\chi_{[y|x]} + \chi_{[\{x,y\}|\emptyset]} = \chi_{[y|\emptyset]}$, $\chi_{[x|y]} + \chi_{[\emptyset|\{x,y\}]} = \chi_{[\emptyset|y]}$ and $\chi_{[\emptyset|\emptyset]} = 1$.
- The inital set of inequalities multipied by all variables $\chi_{[P|N]}$ of size one:
  $(\chi_{[x|\emptyset]} + \chi_{[y|\emptyset]} \geq 1)\chi_{[x|\emptyset]} = \chi_{[x|\emptyset]} + \chi_{[\{x,y\}|\emptyset]} \geq \chi_{[x|\emptyset]}$,

$(\chi_{[x|\emptyset]} + \chi_{[y|\emptyset]} \geq 1)\chi_{[\emptyset|x]} = \chi_{[y|x]} \geq \chi_{[\emptyset|x]}$,
$(\chi_{[x|\emptyset]} + \chi_{[y|\emptyset]} \geq 1)\chi_{[y|\emptyset]} = \chi_{[\{x,y\}|\emptyset]} + \chi_{[y|\emptyset]} \geq \chi_{[y|\emptyset]}$,
$(\chi_{[x|\emptyset]} + \chi_{[y|\emptyset]} \geq 1)\chi_{[\emptyset|y]} = \chi_{[x|y]} \geq \chi_{[\emptyset|y]}$,
$(\chi_{[x|\emptyset]} \geq \chi_{[y|\emptyset]})\chi_{[x|\emptyset]} = \chi_{[x|\emptyset]} \geq \chi_{[\{x,y\}|\emptyset]}$,
$(\chi_{[x|\emptyset]} \geq \chi_{[y|\emptyset]})\chi_{[\emptyset|x]} = 0 \geq \chi_{[y|x]}$,
$(\chi_{[x|\emptyset]} \geq \chi_{[y|\emptyset]})\chi_{[y|\emptyset]} = \chi_{[\{x,y\}|\emptyset]} \geq \chi_{[y|\emptyset]}$ and
$(\chi_{[x|\emptyset]} \geq \chi_{[y|\emptyset]})\chi_{[\emptyset|y]} = \chi_{[x|y]} \geq 0$.

When the SA rank is k there are $1+\sum_{i=0}^{k}\binom{n}{i+1}2^{i+1}$, different variables in total and at most polynomially times as many inequalities. Since we know linear programming over a polytope is in P, if a given set of inconsistent inequalities requires at most constant SA rank, we can be sure that there exists an algorithm based on the SA operator that can prove them to be inconsistent in polynomial time.

Suppose that the SA rank k formulation of some unsatisfiable ILP instance specifies the empty polytope, i.e. it is an inconsistant linear program. Although this may contain an exponential number of inequalities, not all these may be required to specify the empty polytope. We therefore consider an SA proof to be a subset of these inequalities that specify the empty polytope. The rank of an SA proof is defined to be the size of the largest SA variable in it; the size of an SA proof is simply the number of inequalities in it. The SA rank (size) of an unsatisfiable ILP instance is the smallest possible rank (size) of an SA proof of it.

It can be observed that the SA rank of Horn formulae (see section 2.4.7) and 2CNF formulae is 0 and 1 respectively, as demonstrated in Chapter 3, whilst the SA rank of any polytope arising from an unsatisfiable set of clauses is at most $n-1$ [63].

2.4 Families of Tautologies/Contradictions

In this section we describe some of the most well studied families of tautologies (which we will convert to contradictions) and review some of the bounds known for proofs of them. We are only interested in infinite families (collections) of tautologies, which are defined by some parameters. This is because they allow us to study the

rate of growth in size or rank of potential proofs of the tautologies in various systems against the parameters which define them.

2.4.1 The Pigeonhole Principle

The Pigeonhole Principle (PHP) is the most well studied combinatorial principle in proof complexity. Its popularity is due to the fact that it can easily be formulated and understood, but is generally difficult for proof systems to prove. PHP states that for any two natural numbers n and m where $m > n$, if you tried to put m pigeons into n holes, you would have a hole with more than one pigeon in it. Generally the closer m is to n the harder it is for a system to prove, intuitivly this makes sense because as m gets larger the "more true" it becomes. The worst case where $m = n + 1$ was the first known example of a family of tautologies requiring a Resolution refutation of size $2^{\Omega(n)}$ [36]. However, the so called weak pigeonhole principle, where m is larger than n by more than a constant is more difficult to prove lower and upper bounds on. For recent developments and further information on the Resolution complexity of the weak pigeonhole principle, the reader is referred to [54]. It has been shown that there are polynomially sized proofs of PHP in LS [35] and hence the same holds for SA [48] and LS_+ and also for CP [58], however all these systems, bar LS_+ [35] require at least logarithmic rank to prove it [57, 58].

PHP can be stated as a collection of two sets of clauses, which we will call the *holeset* and the *pigeonset*. Note that in the following definition we can think of the variable $P_{(i,j)}$ as the answer to the question "does pigeon i go to hole j?". The *holeset* states that any two pigeons can not be assigned to the same hole, and consists of all the inequalities of the form $\neg P_{(i,j)} \vee \neg P_{(i',j)}$, where $i \neq i'$, $1 \leq i, i' \leq m$ and $1 \leq j \leq n$. The *pigeonset* states that each pigeon must go to a hole, and is written as the set of clauses $\bigvee_{j=1}^{n} P_{(i,j)}$ for every i, where $1 \leq i \leq m$. A well studied variant of this principle, the so-called Functional Pigeonhole Principle (FPHP), is formulated as the PHP with the additional constraints $\sum_{j=1}^{n} P_{(i,j)} \leq 1$ for each pigeon $1 \leq i \leq m$. This version is clearly more restrictive than the original version and therefore should typically be easier to prove, however it remains hard (i.e. super-polynomial regardless of the value of m) for general Resolution [55]. For a survey

of proof complexity results on the various versions of the Pigeonhole Principle see [56].

2.4.2 The Least Number Principle

The Least Number Principle (LNP), also known as the Minimum Element Principle or the Well-ording Principle, is another well studied combinatoric principle and states that every set of $n \geq 1$ natural numbers has a smallest element. This principle is one of the interesting examples having only exponentially sized Tree-like Resolution proofs but polynomially sized general Resolution proofs as shown in [11]. It is known to require linear rank Resolution, CP and LS_0 proofs, as shown in [12]. When discussing this principle we can consider the variable $\chi_{(i,j)}$ to reflect the answer to the question "is the ith element of the set smaller than the jth element?". The principle can be written as a conjunction of three sets of clauses which we will refer to as *trans*, *lower* and *self*. The set *trans* is so called because it ensures that the transitive property of the set elements is adhered to, namely that if $i \prec j$ and $j \prec k$ then $i \prec k$ must be true; this translates into the set of clauses $\neg\chi_{(i,j)} \vee \neg\chi_{(j,k)} \vee \chi_{(i,k)}$ for all i,j and k, where $1 \leq i, j, k \leq n$. The set of clauses *lower* take the form $\bigvee_{i=1}^{n} \chi_{(i,j)}$ for all j, where $1 \leq j \leq n$, and state that there must be at least one element of the set smaller than the jth one. The final set *self* states that an element is not smaller than itself, this translates to the set of single literal clauses $\neg\chi_{(i,i)}$ for all i, where $1 \leq i \leq n$. The LNP is known to require SA and LS rank precisely $n-2$ [23].

2.4.3 The Tseitin Tautologies

The Tseitin Tautologies, also known as the Tseitin Graph Tautologies in the literature, are the first examples of tautologies proven hard for Regular Resolution by Tseitin in 1968 [65]. The idea behind these tautologies is the fact that not every vertex of an odd-sized graph (i.e. a graph with an odd number of vertices), can have an odd degree. This is trivially true since each edge in a graph adds two to the sum of the degrees in the graph, and hence the sum of the degrees must be even.

The Tseitin tautology of a given odd-sized graph $G = (V, E)$ is defined as follows: it has a boolean variable x_{uv} for each edge $(u, v) \in E$, and has a number of clauses representing the mod 2 equalities

$$\sum_{v \in \Gamma(u)} x_{uv} \equiv 1 \pmod 2,$$

for each $u \in V$, where $\Gamma(u)$ refers to the neighborhood of u. To represent the above mod 2 equalities we need an exponential number of clauses in the size of the neighborhood of u, and as a result typically only families of Tseitin tautologies generated from set of d-regular graphs are studied, where d is a constant. In particular, usually only sets of d-regular expander graphs (sets of graphs with high edge-expansion, see [38]) are considered, since it is the expansion property that is used to derive the lower bounds. For examples of proofs of lower bounds on Tseitin formulae of these graph types, see [12] for linear lower bounds on the CP and LS_+ rank, and [66] for size lower bounds for general Resolution for these formulae.

2.4.4 The House Sitting Principle

Imagine a street on which the houses are numbered 1 to n. The houses get better as you go up the street, with house 1 being the worst shack and house n being the best mansion. Just for fun, the tenants decide they want to swap houses for a week, but, as they all want a good deal, they are unwilling to swap to a house worse than their own, except the greedy lot in house 1 who are unwilling to stay even in their own house. Each house is big enough to accommodate all the tenants of the street, but no-one wants to stay in any house that the owners are staying. The House-Sitting Principle (HSP) states that in such a situation, one can never satisfy all the tenants of the street. Its negation is formulated using variables $p_{(i,j)}$ which can be taken to be the answer to the question, "do the tenants from house i go to house j?". In propositional form it is stated as the following set of clauses: $\bigvee_{j=i}^{n} p_{(i,j)}$ for all $1 \leq i \leq n$ (all tenants go to a house at least as good as their own), $\neg p_{(i,j)} \vee \neg p_{(j,j)}$ for all $1 \leq i, j \leq n$ where $j > i$ (if the owners are in the house other people can't stay there) and $\neg p_{(1,1)}$ (the owners of house 1 do not stay in their own house. This principle is an example of a family of tautologies with large clauses (up

to n variables), but is very easy to prove for all studied proof systems, since it can be refuted using unit propagation and is inconsistent when expressed as a linear program.

2.4.5 Random kCNF formulae

As well as the complexity of SAT instances arising from combinatorial principles, the complexity of randomly generated kCNF formulae (particularly for $k = 3$) have also been studied in proof complexity. These formulae were introduced in [14] and are created by taking a subset of m clauses uniformly at random from the set of all $2^k\binom{n}{k}$ possible clauses of length at most k. In this work it is shown that there exists a value Δ_k, depending only on k, where random instances of kSAT with more than $\Delta_k n$ clauses become almost certainly unsatisfiable. It is an important open question, known as the sharp-threshold conjecture, whether there exists for each k a constant c_k such that the following holds: for each positive constant ϵ, if $m \geq c_k n + \epsilon$ (resp. $m \leq c_k n - \epsilon$) then a randomly chosen instance of kSAT with m clauses is unsatisfiable (resp. satisfiable) with high probability. It is known that $c_2 = 1$ [33] and c_3, if it exists, is between 3.003 and 4.758 (see [30] and [42] respectively). It is also known that c_k must be at most $2^k n \ln 2$, as shown in [14]. It is also shown in [14] that any Resolution proof of the unsatisfiability of randomly generated (unsatisfiable) instances with $O(n)$ clauses requires exponential size with high probability. It was later shown in [5] that unsatisfiable 3SAT instances with at most $O(n^{\frac{6}{5}-\epsilon})$ clauses also require exponential size Resolution proofs with high probability. One motivation for studying such instances is that it is likely that these instances will be difficult for any proof system to solve, when m is set to a suitable value (i.e. near the threshold between where instances go from being almost certainly satisfiable to almost certainly unsatisfiable).

The downside of studying these formula is that they are relatively difficult to prove bounds on, since there is little structure to work with. Riis in [61] notes that it is no accident that the best known lower bounds in proof complexity come from highly structured instances, rather than random ones. Currently proofs about the complexity of random kCNF formulae are based on the fact that with high prob-

ability, the instances have high *expansion*. There are several notions of expansion; the expansion of a set of clauses X is $\epsilon(X) = N(X)/|X|$, where $N(X)$ is the number of variables which appear in these clauses X. If this value is high for all subsets of a given size, the instance is said to have high expansion. Further information on expansion is available in [12, 38].

The rank complexity of random kCNF formulae for $k \geq 5$ has recently been studied for Cutting Planes and the LS systems in [12], whereas the authors prove that the rank complexity of such instances is linear with high probability if one chooses an appropriate number of clauses (some constant, dependant on k, times the number of variables). One of the tools used to prove these results is the fact that one can consider random kCNF formulae to be subformulae of randomly selected mod 2 inequalities on k variables. A mod 2 formula on k variables is one of the form $\sum_{i \in S} x_i \equiv a \pmod 2$ where $S \subseteq [n]$, $|S| = k$, and $a \in \{0, 1\}$; each such formula can be represented as a conjunction of $2^k - 1$ clauses. By considering CNF formulae generated using randomly selected mod 2 formulae, as opposed to random kCNF formulae, one has more structure to work with, simplifying proofs of complexity.

2.4.6 Ramsey's Theorem

Ramsey's theorem is an interesting result about graphs. In its simplest form it states that any undirected simple graph (i.e. one with no multiple edges or self-loops) on n vertices has either a clique or an independent set of size $\lfloor \frac{\log n}{2} \rfloor$. A proof of this theorem can be found in [41]. The propositional version, RAM^n, has variables x_e for all possible edges $e \in [n]^2$, and can be stated as the set of clauses:

$$\bigvee_{e \in [X]^2} x_e \text{ and } \bigvee_{e \in [X]^2} \neg x_e$$

for all sets X of size $\lfloor \frac{\log n}{2} \rfloor$ on the elements $1, \ldots, n$.

This infinite family of contradictions is interesting to study in Proof Complexity due to the fact that they are particularly difficult to argue about. In particular, the Resolution complexity of RAM^n is currently unknown, despite the fact that we know the Resolution width of an RAM^n is at least $\frac{1}{2} n^{\frac{1}{4}}$, as shown in [46]. RAM^n

represents a case for which known techniques for proving lower bounds on the size of Resolution proofs seem to fail. However as a result of [46], the Resolution complexity of RAM^n is known to be linked to the complexity of the weak pigeonhole principle where you have n^4 pigeons to n holes, namely that if the latter requires exponential size Resolution proofs then so must the former. Little is known about the complexity of RAM^n for methods based on ILP.

One motivation behind researching the complexity of RAM^n is that it could potentially further our understanding of random graphs and lead to improved bounds on the Ramsey numbers $R_{n,n}$. These numbers, which have been extensively studied in graph theory, are defined as the minimum number of vertices required in a graph to guarantee the existence of a clique or independent set of size n. For instance $R_{3,3} = 6$. However, despite much research, we do not even know the exact value of $R_{n,n}$ for $n \geq 5$ [53].

2.4.7 HornSAT and 2SAT

HornSAT and 2SAT are two well understood restrictions of the SAT problem studied in proof complexity. HornSAT is the problem of finding out whether a set of Horn clauses are satisfiable, where a Horn clause is one with at most one positive literal and any number of negative ones. 2SAT asks whether there is a satisfying assignment for a set of clauses, where each clause has at most 2 literals. Both these problems are known to be solveable in polynomial time. This is because any unsatisfiable HornSAT instance can be refuted using unit propagation and any unsatisfiable 2SAT instance can be refuted by considering each pair of variables and seeing whether all four possible assignments of 0/1 values to them contradict at least one of the clauses. The reason for studying instances of these two problems in proof complexity is to demonstrate that a particular proof system can solve them easily. As a general rule, any proof system which can not easily refute such instances, or any complexity measure which is not constant for all such instances, is of little interest. In particular both Horn and 2CNF formulae require constant narrow Resolution width, and constant rank proofs in all the systems mentioned in this thesis.

2.5 Size and Rank Maps of Various Proof Systems

The current known size separations between the proof systems mentioned in this review are given in Figure 2.3. In this figure, systems on the right of the central line are those which are known to have exponential size lower bounds. The references given in the diagram indicate the work in which the separation is proven, references next to proof systems indicate the works in which exponential size lower bounds are shown for that system. Note that "TLResolution" refers to Tree-like Resolution. In this figure there are several types of edges, the meaning of which is as follows:

$\longrightarrow$ the source is provably stronger than the target (i.e. the source p-simulates the target and there are examples which require only polynomial sized proofs in the source but exponential sized proof in the target).

$\cdots\!\!\succ$ the source p-simulates the target but there are no examples known that separate them.

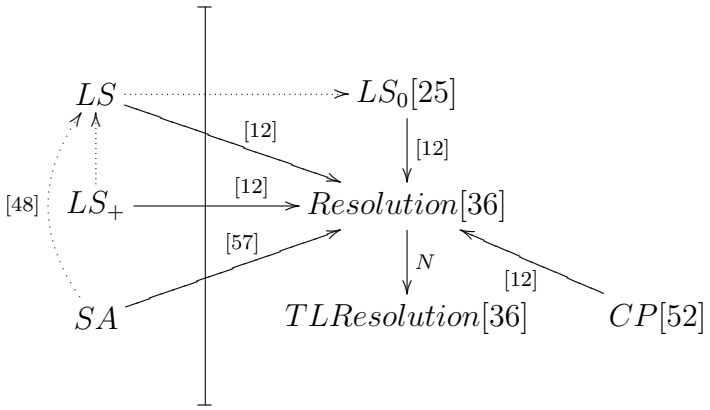


Figure 2.3: Known relationships between the sizes of proofs in various systems.

One might expect that all these proof systems have examples for which they require exponentially sized proofs and that examples can be found that separate the size of proofs in SA, LS, LS_0 and LS_+. The definition of the systems and known rank bounds suggest that the most likely separation between the other systems is that SA is incomparable to LS_+ and that CP is incomparible to any of the other

systems.

The current known separations between the rank complexity of the systems mentioned in this paper, updated with the results of this thesis, are presented in Figure 2.4. In this figure we also include the complexity measure of narrow Resolution width (NResWidth). Although this is not strictly a measure of proof rank, it is an interesting and a relevant complexity measure in the context of the other systems. The meaning of the arrows used in Figure 2.4 is defined as follows:

$\longrightarrow$ the source is provably stronger than the target (i.e. there are examples in which the rank complexity of the source can be arbitrarily smaller than the rank complexity of the target, and the source rank-simulates the target).

$\cdots\!\!\rightarrow$ the source rank-simulates the target but there are no examples known that separate them.

$\rightsquigarrow$ there are examples in which the rank complexity of the target is arbitrarily smaller than the source, but no examples are known the other way, nor is it known if the source rank-simulates the target.

$\leftsquigarrow\!\!\rightsquigarrow$ the two systems at either end of the edge are known to be incomparable with respect to rank complexity (i.e. there are families of tautologies requiring proofs of asymptotically larger rank in one system than in the other and vice versa).

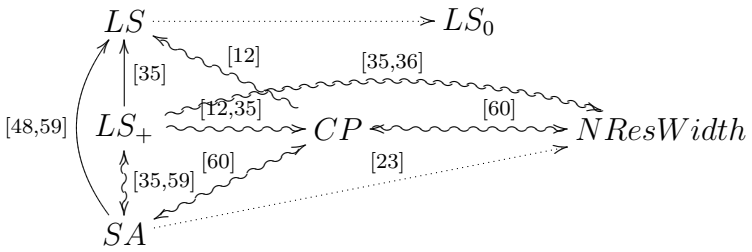


Figure 2.4: Known relationships between the rank complexity of various proof systems.

It should be noted that further separations are known, such as the fact that LS_+ and Cutting Plane rank are incomparable, as shown in [19], however these separations come from instances not derived from unsatisfiable CNF formulae.

Chapter 3

Rank Bounds for the SA Proof System

3.1 Introduction

In this chapter we consider the Sherali-Adams (SA) operator as a proof system and prove linear lower bounds on the SA rank required to prove both the Pigeonhole Principle (PHP) and the Least Number Principle (LNP). We also present short proofs that the SA rank of any unsatisifiable Horn formula is zero, whilst the rank of any unsatisfiable 2CNF formula is one. In addition we demonstrate the system is both sound and complete, the latter holding as a result of a simulation of (general) Resolution. We also define the size of an SA proof and show that while the two principles require linear rank, they only require at most polynomially sized proofs, thereby showing that proofs in the system can not be balanced (i.e. if there is a proof of size s, there does not necessarily exist a proof of rank $\log s$), unlike Frege proofs (see [45]).

3.1.1 Related Work

Although the SA operator has been around since 1990, and has been studied a number of times as a method for solving ILP (e.g. see [8, 48]), little is known about its power as a proof system. The rank lower bounds presented in this chapter are

the first linear rank lower bounds proven for the SA proof system and were initially presented in [57]; the other results presented in this chapter come from a follow-up work, [23]. The only other work presenting rank bounds for the SA operator is [22], in fact it is a consequence of the dichotomy theorem of that work that the SA ranks of both PHP and LNP are non-constant; in fact $\Omega((\log n)^c)$, for some constant $c \geq 0$. It also follows from [63, 48] that the SA ranks of PHP and LNP are at most $n^2 + n - 1$ and $n^2 - 1$ respectively (one less than the number of variables in the inequalities). In this chapter we establish that the SA ranks of PHP and LNP are at least $\frac{n-2}{2}$ and at least $n - 2$ respectively; the latter is known to be an exact bound due to the matching upper bound presented in [23]. This work also includes an improved and perfected bound for PHP, which is accomplished through a model counting argument, in a similar fashion to the dichotomy theorem of [22]. In this chapter however, we present the original proof that PHP requires linear SA rank, since it is intuitive and the author can lay claim to having produced it.

Although as a corollary of our demonstration that SA simulates Resolution we have that SA is complete as a proof system, this was already known as it is a consequence of the results presented in the paper by Sherali and Adams [63], in which the system was defined. What is interesting about the proof presented here however is that it shows that the SA rank of any polytope derived from an unsatisfiable CNF formula F is less that or equal to the Resolution width of F. This property was subsequently proved in [59] not to hold for the well known LS_+ and CP systems; these results are presented in Chapter 5.

3.2 Soundness and Completeness

We consider the SA operator to be defined as in Section 2.3.3. As an introduction to the SA operator, we will begin by proving that as a proof system it is both sound and complete (see page 4).

Let S_0^F be a polytope defined by converting a CNF F on n variables as in Section 2.3, and let S_k^F be the rank k SA formulation of F (as in section 2.3.3). To see that the SA proof system is sound, we will demonstrate that it is not possible to derive

the empty polytope using the SA operator if the initial CNF formula is satisfiable. More specifically, we will show that if F is satisfiable then S_k^F is non-empty, for all $0 \leq k \leq n-1$ ($n-1$ being the maximum possible SA rank [63]).

Proposition 3.2.1 If F is a satisfiable CNF formula on the variables $x_1, \ldots, x_n$, then the rank k SA formulation of F (S_k^F) is non-empty, for all $0 \leq k \leq n-1$.

Proof: It suffices to show that S_{n-1}^F is non-empty since this is defined by a super-set of the equalities and inequalities defining S_j^F for all $0 \leq j \leq n-2$. Let $y^0 = (\chi_{[x_1|\emptyset]}, \ldots, \chi_{[x_n|\emptyset]}, \chi_{[\emptyset|x_1]}, \ldots, \chi_{[\emptyset|x_n]}, \chi_{[\emptyset|\emptyset]})$ be a 0-1 point vector in S_0^F. We can guarantee that at least one such point exists since the satisfying assignments of F correspond directly to the integer solutions of S_0^F and F has at least one satisfying assignment. Now consider the point vector y^{n-1} whose entries are given by

$$\chi_{[x_{i_1} \cup \ldots \cup x_{i_t} | x_{j_1} \cup \ldots \cup x_{j_f}]} := \chi_{[x_{i_1}|\emptyset]} \cdot \ldots \cdot \chi_{[x_{i_t}|\emptyset]} \cdot \chi_{[\emptyset|x_{j_1}]} \cdot \ldots \cdot \chi_{[\emptyset|x_{j_f}]},$$

where $t + f \leq n-1$. To see that S_{n-1}^F is non-empty we will argue that it contains y^{n-1}.

Recall from Section 2.3.3 that S_{n-1}^F is defined by (1) the inequality $\chi_{[\emptyset|\emptyset]} = 1$, (2) the *bounding inequalities*, (3) the *equalities of negation* and (4) the set of inequalities derived from the clauses of F, multipied by all possible SA variables of size $n-1$. The point y^{n-1} trivially satisifes (1) and since it is a 0-1 vector, must also satisfy (2).

It is also not difficult to see that y^{n-1} must also satisfy the equalities of negation. Consider one such equality: $\chi_{[P \cup i|N]} + \chi_{[P|N \cup i]} = \chi_{[P|N]}$. From the definition of the point S_{n-1}^F this is equivilent to $\chi_{[P|N]} \cdot \chi_{[i|\emptyset]} + \chi_{[P|N]} \cdot \chi_{[\emptyset|i]} = \chi_{[P|N]}$. Since the values of $\chi_{[i|\emptyset]}$ and $\chi_{[\emptyset|i]}$ are the same in both y^0 and y^{n-1}, and their values in y^0 correspond to a valid assigment to the literals x_i and $\neg x_i$ respectively in F, we can be sure that exactly one of $\chi_{[\emptyset|i]}$ and $\chi_{[i|\emptyset]}$ is 0 whilst the other is 1 in y^{n-1}. Hence, the equalities of negation defining S_{n-1}^F are satisfied by y^{n-1}.

The equalities of (4) are also satisfied by y^{n-1}, to see this, consider one such inequality: $\chi_{[P \cup i_1|N]} + \ldots + \chi_{[P \cup i_t|N]} + \chi_{[P|N \cup j_1]} + \ldots + \chi_{[P|N \cup j_f]} \geq \chi_{[P|N]}$. If the variable $\chi_{[P|N]} = 0$ in y^{n-1} then the inequality is trivially satisfied. If on the other hand $\chi_{[P|N]} = 1$, then it must be that at least one of the variables on the left hand

side of the inequality equals 1 in y^{n-1} too; hence satisfying it. From the definition of the point y^{n-1}, the inequality is equivilent to $\chi_{[i_1|\emptyset]} \cdot \chi_{[P|N]} + \ldots + \chi_{[i_t|\emptyset]} \cdot \chi_{[P|N]} + \chi_{[\emptyset|j_1]} \cdot \chi_{[P|N]} + \ldots + \chi_{[\emptyset|j_f]} \cdot \chi_{[P|N]} \geq \chi_{[P|N]}$, which, since $\chi_{[P|N]} = 1$ simplifies to $\chi_{[i_1|\emptyset]} + \ldots + \chi_{[i_t|\emptyset]} + \chi_{[\emptyset|j_1]} + \ldots + \chi_{[\emptyset|j_f]} \geq 1$. This is precisely one of the defining inequality of S_0^F, which we know to contain y^0. Since y^{n-1} has the same values for all the variables which appear in this inequality as y^0, it must be that it is also satisfied by y^{n-1}.

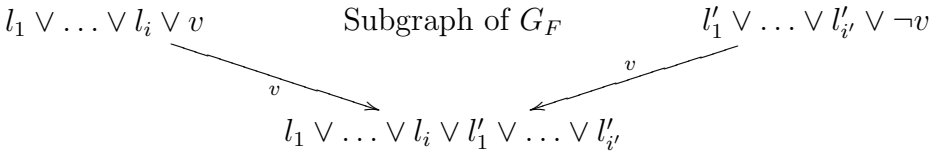
□

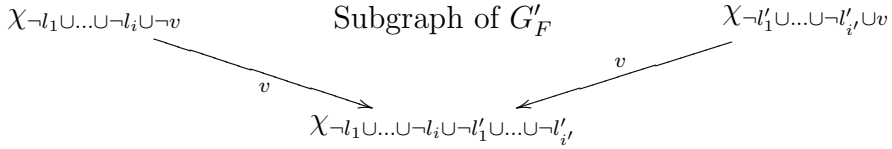
Completeness with respect to the SA proof system means that if F is contradictory, then there exists a value $0 \leq k \leq n-1$, such that S_k^F is the empty polytope. We will argue completeness by simulation of Resolution, although the result follows from [63, 48]. As mentioned in Section 2.2, a Resolution refutation may be seen as a directed acyclic graph (DAG) from the initial clauses to the empty clause. In this DAG each non-source node has two parents from which it was deduced via the Resolution rule (as described in Section 2.2). Note that to simplify the notation, in the following proposition we write the SA variable $\chi_{[P|N]}$ as $\chi_{P_1 \cup \ldots \cup P_{|P|} \cup N_1 \cup \ldots \cup N_{|N|}}$.

Proposition 3.2.2 If F is a contradictory CNF whose Resolution width is k, then S_0^F has SA rank $\leq k$.

Proof: It suffices to prove that S_k^F is empty. Consider a Resolution DAG G_F for F of width k. Each node in G_F is labeled by either an initial clause or by $(l_1 \vee \ldots \vee l_i \vee l'_1 \vee \ldots \vee l'_{i'})$ with two parent nodes $(l_1 \vee \ldots \vee l_i \vee v)$ and $(l'_1 \vee \ldots \vee l'_{i'} \vee \neg v)$, where v is a variable and $l_1, \ldots, l_i, l'_1, \ldots, l'_{i'}$ are literals of F, and $i+1 \leq k$, $i'+1 \leq k$ and $i + i' \leq k$. Consider the related DAG G'_F in which these labels are substituted

by $\chi_{\neg l_1 \cup \ldots \cup \neg l_i \cup \neg l'_1 \cup \ldots \cup \neg l'_{i'}}$ and $\chi_{\neg l_1 \cup \ldots \cup \neg l_i \cup \neg v}$ and $\chi_{\neg l'_1 \cup \ldots \cup \neg l'_{i'} \cup v}$.





Each source in G'_F corresponds to an initial clause, and has the label $\chi_{\neg l_1 \cup \neg l_2 \ldots \cup \neg l_r}$ if $(l_1 \vee l_2 \ldots \vee l_r)$ is the corresponding clause of F. The unique sink of $G'_F$is labeled $\chi_\emptyset$. If we begin with the inequality from the converted clause $\chi_{l_1} + \chi_{l_2} + \ldots + \chi_{l_r} \geq 1$, then multiply this by $\chi_{\neg l_1 \cup \ldots \cup \neg l_r}$, using the equalities of negation we obtain

$$0 \geq \chi_{\neg l_1 \cup \neg l_2 \cup \ldots \cup \neg l_r}$$

as a consequence of S_k^F (since $r \leq k$). We now proceed by induction on the distance from a sink in G'_F, to prove that every node χ_D must take a value at most 0. Since the unique sink is labeled by $\chi_\emptyset$, this will contradict the SA axiom $\chi_\emptyset = 1$. The base case, distance 0 from a source, has been proved. Suppose it is true for distance $\leq m$. Consider a node labeled $\chi_{\neg l_1 \cup \ldots \cup \neg l_i \cup \neg l'_1 \cup \ldots \cup \neg l'_{i'}}$ at distance $m+1$. Its parents are labeled $\chi_{\neg l_1 \cup \ldots \cup \neg l_i \cup \neg v}$ and $\chi_{\neg l'_1 \cup \ldots \cup \neg l'_{i'} \cup v}$, and both must be evaluated to at most 0 by the inductive hypothesis. It follows from monotonicity (see page 23) that $\chi_{\neg l_1 \cup \ldots \cup \neg l_i \cup \neg l'_1 \cup \ldots \cup \neg l'_{i'} \cup \neg v}$ and $\chi_{\neg l_1 \cup \ldots \cup \neg l_i \cup \neg l'_1 \cup \ldots \cup \neg l'_{i'} \cup v}$ must both be at most 0 (and since $i + i' \leq k$ the inequalities with these variables are both present in the kth lift), whereupon $\chi_{\neg l_1 \cup \ldots \cup \neg l_i \cup \neg l'_1 \cup \ldots \cup \neg l'_{i'}} \leq 0$ follows from the equalities of negation. □

Corollary 3.2.1 If F is a contradictory CNF whose Resolution width and size are k and s respectively, then S_0^F has SA size at most $(k+1)s + 1$.

Proof: In order to derive each of the source inequalities of the form $0 \geq \chi_{\neg l_1 \cup \neg l_2 \cup \ldots \cup \neg l_r}$, we require $r + 1 \leq k + 1$ inequalities: the inequality formed by multiplying $(\chi_{l_1} + \ldots + \chi_{l_r} \geq 1)$ by $\chi_{\neg l_1 \cup \ldots \cup \neg l_r}$ together with r equalities of negation. During each simulation of a Resolution step, we require $i + i' + 1 \leq k + 1$ inequalities: one equality

of negation and $i + i'$ inequalities to derive monotonicity. Finally, at the sink, we require the extra equality $\chi_\emptyset = 1$. □

3.3 The SA rank of Horn and 2CNF formulae

As mentioned in Section 2.4.7, HornSAT and and 2SAT are restricted forms of the SAT problem in which the input formulae are restricted to a conjunction of Horn clauses and a conjunction of 2-clauses respectively. In this section we show that a polytope derived from an unsatisfiable Horn formula requires SA rank 0 whilst a polytope derived from an unsatisfiable 2CNF formula requires SA rank 1. This demonstrates that even with constant rank, there are still entire families of instances which can be proven using the SA proof system.

Proposition 3.3.1 The SA rank of an unsatisfiable Horn formula F is 0.

Proof: The SA rank 0 formulation of F is simply the linear program produced by converting F as described in Section 2.3. Hence it suffices to show this linear program is contradictory. To see this, note that F must contain at least one single literal clause, otherwise F could be satisfied by setting all variables negatively. If one applies unit propagation to F (i.e. the rule $A \vee l$ and $\neg l$ gives A, for any literal l and any disjunction A), then the resulting formula must also have single literal clauses, for the same reason. One can see that eventually, using the same logic, applying unit propagation will result in deducing the empty clause. In the linear programming setting, applying unit propagation is equivalent to simply adding the inequalities defining the clauses $\neg l$ and $A \vee l$ together. Eventually by adding enough times, one will be able to deduce the contradiction $0 \geq 1$. □

Proposition 3.3.2 The SA rank of an unsatisfiable 2CNF formula F is 1.

Proof: For convenience we use the notation $\chi_{[P|N]} := \chi_{P_1 \cup \ldots \cup P_{|P|} \cup N_1 \cup N_{|N|}}$. Let G_F be the directed graph whose vertex set $V(G_F)$ contains exactly one vertex for each literal of F and whose edge set $E(G_F)$ contains the edges $\neg xy$ and $\neg yx$ for each clause $x \vee y$ of F. One can consider an edge to mean that if the source is true, the sink must be true in order for the formula to be satisfied. It is well known (and easy

to see) that F is unsatisfiable if and only if G_F has a path between x and $\neg x$ and $\neg x$ and x for some variable x. For each path $p_1 p_2 \dots p_i$ in this graph, where $p_1 = x$ and $p_i = \neg x$ for some variable x, one can see that multiplying the inequalities that define this path by the literal x, adding them together and substituting using the equalities of negation, enables us to derive the inequality $(i-2)\chi_{\neg x} \geq (i-1)\chi_{\neg x}$. For instance, for the path $xyz\neg x$, we may have:

$$\begin{gathered}
(\chi_{\neg x} + \chi_y \geq 1) \times \chi_x \\
+(\chi_{\neg y} + \chi_z \geq 1) \times \chi_x \\
+(\chi_{\neg z} + \chi_{\neg x} \geq 1) \times \chi_x \\
= 2\chi_{\neg x \vee x} + \chi_{y \vee x} + \chi_{\neg y \vee x} + \chi_{\neg z \vee x} + \chi_{z \vee x} \geq 3\chi_x. \\
2\chi_x \geq 3\chi_x.
\end{gathered}$$

Since we know there must be a path between some x and $\neg x$ and vice versa in G_F, we know we will be able to derive the inequalities $i\chi_x \geq (i+1)\chi_x$ and $j\chi_{\neg x} \geq (j+1)\chi_{\neg x}$ for some i and j, since we also have that $\chi_x \geq 0$ and $\chi_{\neg x} \geq 0$, we can derive that $\chi_l = 0$ and $\chi_{\neg l} = 0$. However, this contradicts the inequality of negation $\chi_l + \chi_{\neg l} = \chi_\emptyset = 1$. Since we only required variables and inequalities present in the level 1 SA formulation to generate a contradiction, the SA rank of a polytope derived from a given unsatisfisable 2CNF is always 1. □

3.4 Rank Lower Bounds

In this section we prove rank lower bounds for the SA operator on the inequalities defining the well known Pigeonhole and Least Number Principles, which we defined in propositional form in Sections 2.4.1 and 2.4.2 respectively.

Theorem 3.4.1 The Pigeonhole Principle requires at least SA level $\lceil \frac{n+1}{2} \rceil - 2$ to prove.

Proof: To show this statement is true we will argue that the set of inequalities produced by SA level $\lfloor \frac{n}{2} \rfloor - 2$ can be satisfied by assigning suitable values to the variables.

The value we assign to the variable $\chi_{[P|N]}$ is roughly related to the probability of picking a random assignment of $\frac{n}{2}$ pigeons to $\frac{n}{2}$ holes, including all those holes mentioned in the pairs in the variable, and finding that the information declared within the variable fits the assignment. It is calculated as follows:

1. If P contains two pairs (i, j) and (i', j), then we can see straight away there is a contradiction and so assign the value 0.

2. Otherwise for every $1 \leq j \leq n$, where j is not the second element of any pair in P, we take all pairs in N which have j as the second element, and put them in a new set, which we denote by S_j.

3. We assign any variable not already assigned 0 the value $(\frac{2}{n})^{|P|} \times \prod_{j=1}^{n}(1-\frac{2|S_j|}{n})$.

We can see straight away that when the SA level is at most $\lfloor \frac{n}{2} \rfloor - 1$ and $n \geq 2$ all inequalities of the form $\chi_{[P|N]} \geq 0$ and $\chi_{[P|N]} \leq 1$ are trivially satisfied by the method, as is the constraint that $\chi_{[\emptyset|\emptyset]} = 1$ and that if $\chi_{[P|N]} = 0$, then $\chi_{[Q|W]} = 0$, where $P \subseteq Q$ and $N \subseteq W$.

Lemma 3.4.2 The method provides values which satisfy the equations of negation where $|P \cup N| \leq \frac{n}{2} - 1$ and $n > 2$.

Proof: Recall that the equations of negation are $\chi_{[P\cup(i,j)|N]} + \chi_{[P|N\cup(i,j)]} = \chi_{[P|N]}$ for all pigeons i, all holes j and all possible sets P and N, where $(i, j)P \cup N$. To prove this lemma, we have to consider three cases, depending on whether or not the hole j already appears in N or P. In case (1), j does not appear in either N or P, in case (2), j appears in P and finally in case (3), j does not appear in P but does appear at least once in N. Throughout we will denote the value assigned to the variable $\chi_{[P|N]}$ as ξ.

Case (1): In this instance $\chi_{[P\cup(i,j)|N]} = \xi \times \frac{2}{n}$, since the extra pair does not affect the set S_j and therefore the only difference is that the set P is one larger. We can also see that $\chi_{[P|N\cup(i,j)]} = \xi \times \frac{n-2}{n}$. Given this, it is easy to show that such

equations are satisfied for any value of ξ since substituting these values gives us the following:

$$\chi_{[P\cup(i,j)|N]} + \chi_{[P|N\cup(i,j)]} = \chi_{[P|N]}$$
$$(\xi \times \frac{2}{n}) + (\xi \times \frac{n-2}{n}) = \xi.$$

Case (2): In this instance $\chi_{[P\cup(i,j)|N]} = 0$ since we have at least two pairs in P with hole j, however we can be sure that $\chi_{[P|N\cup(i,j)]} = \xi$ as $S_j = \emptyset$, since the hole j appears in P. Using these values, the equation is trivially satisfied.

Case (3): In this situation, $\chi_{[P|N\cup(i,j)]} = \kappa \times (f - \frac{2}{n})$ where $\xi = \kappa \times f$ and $f = \frac{n-(2|S_j|)}{n}$, since the additional pair in N makes S_j one larger. We can also see that $\chi_{[P\cup(i,j)|N]} = \kappa \times \frac{2}{n}$, since the factor f is removed by the addition of the positive assignment of a pigeon to hole j, and the addition makes $|P|$ one larger, hence the inclusion of the factor $\frac{2}{n}$. We can see that this gives us the required result as follows:

$$\chi_{[P\cup(i,j)|N]} + \chi_{[P|N\cup(i,j)]} = \chi_{[P|N]}$$
$$\kappa \times \frac{2}{n} + \kappa \times (f - \frac{2}{n}) = \kappa \times f$$
$$f - \frac{2}{n} + \frac{2}{n} = f$$

□

Lemma 3.4.3 The method satisfies the *holeset* inequalities when SA level $\lfloor \frac{n}{2} \rfloor - 2$ is used and $n \geq 4$.

Proof: To prove this lemma, we will split the set of all possible variables by which the *holeset* inequalities maybe multiplied, when using SA level $\lfloor \frac{n}{2} \rfloor - 2$, into 5 distinct cases and discuss each in turn. Throughout this section we will use δ to refer to the variable by which the inequalities are multiplied. The positive side of this variable will be denoted δ_P and the negative side δ_N. We will denote the value assigned by the method to the variable δ as ξ.

Case 1: $(k, j) \notin \delta_P$ and $(k, j) \notin \delta_N$ for all $1 \leq k \leq n + 1$.

Both $\chi_{[(i,j)|\emptyset]} \times \delta = \xi \times \frac{2}{n}$ and $\chi_{[(i',j)|\emptyset]} \times \delta = \xi \times \frac{2}{n}$ must be true as the addition of these pairs only affects the size of P, as $S_j = \emptyset$. It is easy to see that the inequalities are satisfied when $n \geq 4$, for any possible value of ξ:

$$\begin{aligned} \xi \times \frac{2}{n} + \xi \times \frac{2}{n} &\leq \xi \\ \xi \times \frac{4}{n} &\leq \xi \\ \frac{4}{n} &\leq 1 \end{aligned}$$

Case 2: At least one pair $(i'', j) \in \delta_P$, where $i'' \notin \{i, i'\}$.

This case is trivial since both $\chi_{[(i,j)|\emptyset]} \times \delta$ and $\chi_{[(i',j)|\emptyset]} \times \delta$, contain at least two pairs with the same hole j in their positive parts P and therefore will be set to 0, giving us $0 + 0 \leq \xi$, which is true for all values of $\xi \geq 0$.

Case 3: At least one pair $(i'', j) \in \delta_N$, where $i'' \notin \{i, i'\}$ and $(k, j) \notin \delta_P$ for all k where $1 \leq k \leq n + 1$.

The addition of the pairs (i, j) and (i', j) to δ_P ensures $S_j = \emptyset$, making the inclusion of any pair (i'', j) in δ_N irrelevant. However the addition of these pairs does make P one larger since no other pair in P contains hole j. We define $\xi = \kappa \times f$ where f is the factor $\frac{n-(2|S_j|)}{n}$, giving us $\chi_{[(i,j)|\emptyset]} \times \delta = \chi_{[(i',j)|\emptyset]} \times \delta = \kappa \times \frac{2}{n}$. Using this, together with the fact that the limit on the SA level means $|S_j| \leq \lfloor \frac{n}{2} \rfloor - 2$ we can show the inequality holds for every possible value of ξ:

$$\begin{aligned} \kappa \times \frac{2}{n} + \kappa \times \frac{2}{n} &\leq \kappa \times \frac{n - (2|S_j|)}{n} \\ \kappa \times \frac{4}{n} &\leq \kappa \times \frac{n - (2|S_j|)}{n} \\ \frac{4}{n} &\leq \frac{n - (n - 4)}{n} \end{aligned}$$

Case 4: At least one of $(i, j) \in \xi_P$ and $(i', j) \in \xi_P$ is true.

If both pairs were in ξ_P then all the variables would be 0, so the inequality would be trivially satisfied. If only one of the pairs is in ξ_P then the variable which adds the pair not in ξ_P will contain two pigeons going to the hole j and hence be assigned 0. Since from Lemma 3.4.2 we know adding a pair to a set can never increase the value assigned to the variable, the inequality must be satisfied.

Case 5: At least one of $(i,j) \in \xi_N$ and $(i',j) \in \xi_N$ is true where $(i,j) \notin \xi_P$ and $(i',j) \notin \xi_P$.

From the definition of the SA operator we know that any variable containing the same pair in both positive and negative sides must have the value of 0 (the method does not account for such variables as their value is trivial). In the case where both statements are true, our inequalities are trivially satisfied since we have $0+0 \leq \xi$. In the case that one of the statements is true, we know that one of the variables must be 0, and again from Lemma 3.4.2 we know the other variable can not be assigned a value greater than ξ and hence the inequality is satisfied. □

Lemma 3.4.4 The method satisfies the set of *pigeonset* inequalities, when the maximum SA level is $\lfloor \frac{n}{2} \rfloor - 2$.

Proof: Throughout this proof we denote the variable by which the inequalities can be multiplied by δ. The positive side of δ is denoted δ_P, its negative side δ_N and the value assigned to it by the method is denoted as ξ.

The maximum number of pairs in δ is limited to the the maximum SA level and since each pair contains only one hole, this is also the maximum number of holes that can appear in δ. Each inequality in *pigeonset* contains n different 'j' values or holes. This means that we can be sure that at least $n - (\lfloor \frac{n}{2} \rfloor - 2)$ variables in any inequality in *pigeonset* contain a hole j where $j \notin \delta$. Each of these will be assigned the value $\xi \times \frac{2}{n}$. Therefore, even in the worst case, where all the variables of the form $\chi_{[(i,j)|\emptyset]} \times \delta$ where $j \in \delta$ are assigned the value of 0, any inequality in *pigeonset* is satisfied:

$$(n - (\lfloor \frac{n}{2} \rfloor - 2)) \times (\xi \times \frac{2}{n}) \geq \xi$$
$$(\frac{n}{2} + 2) \times \frac{2\xi}{n} \geq \xi$$
$$\xi + \frac{4\xi}{n} \geq \xi$$

□

Lemmas 3.4.2, 3.4.3, 3.4.4 together show that all the inequalities and equations produced by the SA proof system operating on the Pigeonhole Principle will be satisfied by the values given by the method. Hence we have completed the proof of Theorem 3.4.1.

□

Theorem 3.4.5 The SA rank of LNP is at least $n - 2$.

Proof: We will argue that the set of equations produced by SA level $n - 3$ can be satisfied by assigning suitable values to the variables.

The intuition behind the value we assign a variable is that it is a rough estimate of the probability that, given a random set of n natural numbers, the set's elements match the information contained within the variable. We calculate the value of the variable $\chi_{[P|N]}$ as follows:

1. Consider a DAG which has a vertex for each element of a set of n natural numbers. We add an edge between the ith and jth nodes in this graph if and only if (i, j) appear in P, or if (j, i) appears in N and $i \neq j$.

2. If this graph contains a cycle, possibly consisting of a single edge, there is inconsistent information so we assign the value 0.

3. Remove all edges ij in the graph where there is a longer path between the nodes i and j via some other nodes. This ensures we remove irrelevant information.

4. If the variable has not been assigned the value 0, we assign it the value of $(\frac{1}{2})^E$ where E is the number of edges left in the graph.

For example, the variable $\chi_{[(1,2),(2,3),(1,4)|(4,3),(4,4)]}$ is assigned the same value as the variable $\chi_{[(1,2),(2,3)|(4,3)]}$, namely $\frac{1}{8}$.

It is trivial to show that the method satisfies the constraint $\chi_{[\emptyset|\emptyset]} = 1$, as such a variable will generate a graph with no edges and therefore be assigned $(\frac{1}{2})^0 = 1$. It is also trivial to see that all inequalities $\chi_{[P|N]} \leq 1$ and $\chi_{[P|N]} \geq 0$ for all sets P and N, will be adhered to. It is also easy to see that the inequalities derived from the members of *self* (i.e. $\neg\chi_{(i,i)}$ for all $1 \leq i \leq n$) are also satisfied using this method. In the translation the clause $\neg\chi_{(i,i)}$ becomes the inequality $(-\chi_{[(i,i)|\emptyset]} \geq 1$ and since we assign 0 to any variable containing the pair (i, i) in its positive side we get:

$$(1 - \chi_{[(i,i)|\emptyset]} \geq 1) \times \chi_{[P|N]}$$

$$\chi_{[P|N]} - 0 \geq \chi_{[P|N]}$$

Throughout the rest of this proof we will refer to the variable by which we multiply the inequalities when using the SA operator as $\chi_{[P|N]}$.

Lemma 3.4.6 The inequalities in *lower* are satisfied using this method when the SA level is set to $n - 3$ or less.

Proof: Recall that in CNF form the elements of *lower* are $\bigvee_{i=1}^{n} \chi_{(i,j)}$ for all $1 \leq j \leq n$. These are translated, as in Sections 2.3 and 2.3.3, into the inequalites $\sum_{i=1}^{n} \chi_{[(i,j)|\emptyset]} \geq 1$ for all $1 \leq j \leq n$. As the right hand side of each such inequality is simply 1, we know that this side will take the value of whichever variable we multiply the inequality by. Therefore, when we multiply by the variable $\chi_{[P|N]}$, we get the inequality $\sum_{i=1}^{n} \chi_{[(i,j)\cup P|N]} \geq \chi_{[P|N]}$. If the graph for the variable $\chi_{[P|N]}$ contains a cycle, then so too must each of the variables $\chi_{[(i,j)\cup P|N]}$. This is because these will be represented by the same graph with perhaps one more edge between the ith and jth nodes (if the fact that $i \succ j$ can not be derived from the other edges). In this case the inequality is trivially satisfied as it reads as follows:

$$\sum_{i=1}^{n} 0 \geq 0$$

If however the graph for the variable $\chi_{[P|N]}$ does not contain a cycle, then adding the pair (i, j) will in the worst case create a cycle, and therefore a variable with the

value 0. Apart from the case where $i = j$, which will trivially yeild a loop, the only way this loop can be created by adding the pair (i, j) is if there is a path from j to i; this in turn means that at least one edge must finish at i. For such an edge to exist, at least one of the following must be true: $(k, i) \in P$ or $(i, k) \in N$, for some k, where $1 \leq k \leq n$ and $k \neq i$ as otherwise the variable $\chi_{[P|N]}$ would contain a loop. Since all the variables on the left side of any inequality in *lower* contain different values for i, we can see that even in the worst case we still need at least one pair in the variable $\chi_{[P|N]}$ to render a single variable inconsistent. As SA level $n-3$ allows us at most $n-3$ pairs in $P \cup N$ and each inequality in *lower* contains $n-1$ variables on the left hand side which do not trivially contain a cycle (i.e. all those except the one with the pair (i, i)), we can be sure that at least 2 variables will not contain a cycle. The minimum value these variables can be assigned is half the value of $\chi_{[P|N]}$, in which case the inequality can be shown to be satisfied as follows:

$$\chi_{[(k,j)\cup P|N]} + \chi_{[(k',j)\cup P|N]} \geq \chi_{[P|N]}$$
$$\frac{\chi_{[P|N]}}{2} + \frac{\chi_{[P|N]}}{2} \geq \chi_{[P|N]}$$

Clearly, since this is the worst case, in all other cases where $\sum_{i=1}^{n} \chi_{[(i,j)\cup P|N]}$ has a greater value, the inequalities are also satisfied. □

Lemma 3.4.7 The inequalities in *trans* are satisfied using this method whenever the SA level is $\leq n$.

Proof: In CNF form the elements of *trans* are $\neg\chi_{(i,j)} \vee \neg\chi_{(j,k)} \vee \chi_{(i,k)}$ for all i,j and k, where $1 \leq i, j, k \leq n$; these can be written as the set of inequalities $\chi_{[(i,j)|\emptyset]} + \chi_{[(j,k)|\emptyset]} - 1 \leq \chi_{[(i,k)|\emptyset]}$ for the same values of i, j and k. To prove this lemma, we consider a number of cases. In all but the first case we assume $i \neq j$, $i \neq k$ and $j \neq k$.

Case 1: At least two of i, j or k have the same value. We can see straight away that adding an additional pair to the positive side of a variable can never increase the value assigned to the variable and that any variable which contains a pair (g, g) in its positive side, for any value $1 \leq g \leq n$, is assigned the value 0. Using these

statements and the fact that the left hand side of each inequality in *trans* has just two positive variables we can see that if any of i, j or k have the same value, except when $i = k$ and $i \neq j$, then the inequality is satisfied as one of these positive parts will contain a single edge cycle, take the value 0, and leave us with an inequality in which the single positive part remaining on the left can not be larger than the right side as it is the same variable except it has an additional pair in its positive side.

When $i = k$ and $i \neq j$, the inequality multiplied by a variable becomes $\chi_{[(i,j)\cup P|N]} + \chi_{[(j,i)\cup P|N]} \leq \chi_{[P|N]}$, we can remove the negative part of the left side as it will be given the value 0. This is satisfied by the method as if no path exists between nodes i and j or vice versa in the graph for $\chi_{[P|N]}$ then we get:

$$\frac{\chi_{[P|N]}}{2} + \frac{\chi_{[P|N]}}{2} \leq \chi_{[P|N]}$$

If however it contains one or more of these paths then at least one variable on the left side of the inequality must contain a cycle going from i to j and back again and will therefore be assigned the value 0, in which case the inequality must be satisfied as the remainder of the left is the same as the right with one extra pair in its positive side.

Case 2: The graph for the variable $\chi_{[P|N]}$ contains a cycle. This case is trivial as all the variables on the left must also generate a graph which contains the graph for $\chi_{[P|N]}$ as a subgraph and therefore will also contain a cycle. As all cyclic graphs are assigned the value of 0, the inequality becomes $0 + 0 - 0 \leq 0$, which is trivially true.

Case 3: The graph derived from $\chi_{[P|N]}$ contains a path between nodes i and k. In this case $\chi_{[P\cup(i,k)|N]}$ will generate the same graph, and hence be assigned the same value as the variable $\chi_{[P|N]}$, as the extra pair will be removed as irrelevant. Since adding extra pairs can not increase the value of a variable, even if the other variables are assigned the maximum possible value, the inequality is still satisfied as follows:

$$\chi_{[P\cup(i,j)|N]} + \chi_{[P\cup(j,k)|N]} - \chi_{[P|N]} \leq \chi_{[P|N]}$$
$$\chi_{[P|N]} + \chi_{[P|N]} - \chi_{[P|N]} \leq \chi_{[P|N]}$$
$$1 + 1 - 1 \leq 1.$$

Case 4: The variable $\chi_{[P|N]}$ generates a graph containing neither a path between nodes i and k nor vice versa. In this instance, the variable $\chi_{[P\cup(i,k)|N]}$ will always generate a graph with one more edge than the graph for $\chi_{[P|N]}$ and therefore be assigned half the value of $\chi_{[P|N]}$ or 0 if $\chi_{[P|N]}$ contains contradictory information. The latter instance is already covered by case 2, and the former is also covered as at least one of $\chi_{[P\cup(i,j)|N]}$ and $\chi_{[P\cup(j,k)|N]}$ must be assigned no more than half the value of $\chi_{[P|N]}$ otherwise we would have a path from i to k. Even if the other takes the maximum possible value the inequality is still satisfied:

$$\chi_{[P\cup(i,j)|N]} + \chi_{[P\cup(j,k)|N]} - \frac{\chi_{[P|N]}}{2} \leq \chi_{[P|N]}$$
$$\chi_{[P|N]} + \frac{\chi_{[P|N]}}{2} - \frac{\chi_{[P|N]}}{2} \leq \chi_{[P|N]}$$
$$\chi_{[P|N]} \leq \chi_{[P|N]}.$$

Case 5: The graph for $\chi_{[P|N]}$ contains a path between nodes k and i and does not contain a cycle. When this happens, the variable $\chi_{[P\cup(i,k)|N]}$ will take the value 0. If the graph for the variable $\chi_{[P|N]}$ contains neither a path between nodes i and j nor between nodes j and k then both $\chi_{[P\cup(i,j)|N]}$ and $\chi_{[P\cup(j,k)|N]}$ will take the value of half that of $\chi_{[P|N]}$, thus giving:

$$\frac{\chi_{[P|N]}}{2} + \frac{\chi_{[P|N]}}{2} - 0 \leq \chi_{[P|N]}$$
$$\frac{1}{2} + \frac{1}{2} \leq 1$$

If however $\chi_{[P|N]}$ does contain one of the aforementioned paths, then the variable whose graph contains the path not included in $\chi_{[P|N]}$ will always contain a cycle and

therefore be assigned the value of 0. Even though the other variable left will be set to the same value as $\chi_{[P|N]}$ the inequality is clearly satisfied. □

Lemma 3.4.8 : The method satisfies the equations of negation.

Proof: Recall that the equations of negation are $\chi_{[P\cup(i,j)|N]} + \chi_{[P|N\cup(i,j)]} = \chi_{[P|N]}$ for all sets P and N and all i and j where $1 \leq i, j \leq n$ and $(i, j) \notin P \cup N$. We can show that all these equations are satisfied by considering the following four cases:

Case 1: When the graph associated with $\chi_{[P|N]}$ contains a cycle. This case is trivial since the cycle will also be contained within the graphs of the other variable giving us $0 + 0 = 0$.

Case 2: When $i = j$. In this case we ignore such a pair when it is added to the set N and we set any variable to be 0 when it is added to the set P giving us an equation equivalent to $0 + \chi_{[P|N]} = \chi_{[P|N]}$.

Case 3: When $i \neq j$, the graph for $\chi_{[P|N]}$ is acyclic and does not contain a path between nodes i and j or vice versa. In this case both variables on the left side of the equation will take the value of $\frac{\chi_{[P|N]}}{2}$ because the graph they produce will always contain the extra edge giving us $\frac{\chi_{[P|N]}}{2} + \frac{\chi_{[P|N]}}{2} = \chi_{[P|N]}$.

Case 4: When $i \neq j$ and the graph for $\chi_{[P|N]}$ is acyclic and contains a path between nodes i and j or vice versa. The graph containing the same path as $\chi_{[P|N]}$ will clearly be assigned the same value as it since the extra edge will be removed. The other variable will add an edge which completes a cycle and hence be assigned 0. The equation is then clearly satisfied. □

Since we have proven all the sets of inequalities produced by the SA operator will be satisfied, our proof of Theorem 3.4.5 is concluded. □

3.5 SA Proof Size

Another measure by which we can judge the complexity of SA proofs, rather than the required rank, is the combined number of SA multiplications required to reach a contradiction. This measure we shall refer to as the *size* of the proof. We will now show that for the SA operator, the required rank does not reflect the required proof size; more specifically that the proof required for PHP and LNP is at most polynomial in size, whilst we have already shown the required rank to be linear.

Theorem 3.5.1 There is an SA proof of PHP of polynomial size.

Proof: We will derive the contradiction by deducing both $\sum_{j=1}^{n}\sum_{i=1}^{n+1}\chi_{[(i,j)|\emptyset]} \geq n+1$ and $\sum_{j=1}^{n}\sum_{i=1}^{n+1}\chi_{[(i,j)|\emptyset]} \leq n$. The first inequality can be derived straight away by adding up all the *pigeonset* inequalities, we will prove the other can be derived by induction.

Lemma 3.5.2 From $\chi_{[(i,j)|\emptyset]} + \chi_{[(i',j)|\emptyset]} \leq 1$, we can derive that $\chi_{[(i,j)|(i',j)]} = \chi_{[(i,j)|\emptyset]}$ with a single SA multiplication.

Proof: We accomplish this by multiplying the inequality by the variable $\chi_{[(i',j)|\emptyset]}$ then proceeding as follows:

$$\begin{aligned}
\chi_{[(i,j),(i'j)|\emptyset]} + \chi_{[(i',j)|\emptyset]} &\leq \chi_{[(i',j)|\emptyset]} \\
\chi_{[(i,j),(i'j)|\emptyset]} &\leq 0 \\
\chi_{[(i,j),(i'j)|\emptyset]} &= 0 \\
\chi_{[(i,j),(i'j)|\emptyset]} + \chi_{[(i,j)|(i'j)]} &= \chi_{[(i,j)|\emptyset]} \\
\chi_{[(i,j)|(i'j)]} &= \chi_{[(i,j)|\emptyset]}.
\end{aligned}$$

□

Lemma 3.5.3 We can derive $\sum_{i=1}^{q+1}\chi_{[(i,j)|\emptyset]} \leq 1$ from $\sum_{i=1}^{q}\chi_{[(i,j)|\emptyset]} \leq 1$ with $q+2$ SA multiplications.

Proof: To accomplish this we first multiply by the variable $\chi_{[\emptyset|(q+1,j)]}$ and proceed as follows:

$$\sum_{i=1}^{q} \chi_{[(i,j)|(q+1,j)]} \leq \chi_{[\emptyset|(q+1,j)]}$$

$$\sum_{i=1}^{q} \chi_{[(i,j)|(q+1,j)]} \leq 1 - \chi_{[(q+1,j)|\emptyset]}$$

$$\chi_{[(q+1,j)|\emptyset]} + \sum_{i=1}^{q} \chi_{[(i,j)|(q+1,j)]} \leq 1$$

$$\sum_{i=1}^{q+1} \chi_{[(i,j)|\emptyset]} \leq 1$$

Note that to accomplish the final step we derive $\chi_{[(i,j)|(q+1,j)]} = \chi_{[(i,j)|\emptyset]}$ through applying the method presented in Lemma 3.5.2 to the *holeset* inequality $\chi_{[(i,j)|\emptyset]} + \chi_{[(i',j)|\emptyset]} \leq 1$ for each $1 \leq i \leq q+1$. By Lemma 3.5.2 the final step requires at most $q+1$ SA multiplications. □

By continually applying the method described in Lemma 3.5.3 to each of the inequalities of the form $\chi_{[(1,j)|\emptyset]} + \chi_{[(2,j)|\emptyset]} \leq 1$ in *holeset* we will be able to derive $\sum_{j=1}^{n+1} \chi_{[(i,j)|\emptyset]} \leq 1$ for each value $1 \leq i \leq n+1$. By adding all such inequalities we are able to derive $\sum_{j=1}^{n} \sum_{i=1}^{n+1} \chi_{[(i,j)|\emptyset]} \leq n$. The number of multiplications required to accomplish this was clearly polynomial in n. □

Corollary 3.5.4 There is an SA proof of LNP of polynomial size.

Proof: This follows directly from Corollary 3.2.1 and a result of [31] stating that there is a Resolution refutation of LNP of polynomial size. □

3.6 Open Problems

In this chapter we have shown the SA operator to require linear rank for both PHP and LNP and that both principles only require at most a polynomially sized SA proof. Ultimately, the goal is to prove exponential lower bounds on the size of SA proofs, however this is likely to be difficult, especially since we still know relatively little about the rank complexity of formulas in the SA system. One open question on this front is to find a method to enable us to argue about the SA rank complexity of randomly generated formulae. Whilst similar results are known for Resolution,

Cutting planes and the Lovász and Schrijver systems, as shown in [12], it appears to be more complex to accomplish this for SA.

Chapter 4

The Chvátal Rank of the Pigeonhole Principle

4.1 Introduction

This chapter is concerned with the Cutting Planes proof system (CP), as defined in Section 2.3.1. We demonstrate that the CP rank, also known as the Chvátal rank, of the Pigeonhole Principle (PHP) is $\Theta(\log n)$. In order to prove this we introduce a novel technique which allows us to demonstrate CP rank lower bounds for fractional points with fewer restrictions than previous methods. We also demonstrate that PHP has a polynomially sized CP proof.

4.1.1 Related Work

It is already known that the CP rank of PHP is $O(\log n)$, as shown in [12] and which also follows from an earlier result; Theorem 3.1.1 of [37]. The proofs these papers present were of size $\Omega(n^{\log n})$. We show that there exists a polynomially sized CP proof of PHP. Prior to this work, no non-trivial lower bound on the CP rank of PHP was known. Although Theorem 3.1.1 of [37] proves a logarithmic lower bound on a polytope similar to PHP, its proof relies on the fact that the integer hull of the polytope is non-empty, something which does not hold for PHP. The rank of PHP for other systems based on manipulating linear inequalities is well studied; in [35]

it is shown that the rank of PHP in the standard lift and project system, devised by Lovász and Schrijver in [49], is $n-2$, whilst the same system enhanced with semi-definite cuts can refute PHP in rank 1. A linear lower bound on the PHP rank for the system devised by Sherali and Adams in [63], is given in [57].

Prior to this result the only known CP rank lower bounds for polytopes derived from unsatisfiable CNF formulas, were derived using the technique presented in [12]. This method is somewhat restrictive since it can only be used to show that points whose coordinates are in $\{0, 1/2, 1\}$ can survive a number of rounds of cuts. The work in this chapter demonstrates how one can argue that points consisting of a wide range of fractions can survive a number of rounds.

4.2 Preliminaries

The CP proof system can be considered as a refutation proof system operating on linear inequalities (i.e. it derives the contradiction $1 \leq 0$) which has the axioms $x_i \leq 1$ and $x_i \geq 0$ for any variable x_i and the following inference rule, which we will call the cut rule:

$$\frac{\begin{array}{c} a_{11}x_1 + \cdots + a_{1n}x_n \geq b_1 \\ \cdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n \geq b_m \end{array}}{(\sum_{i=1}^{m} \lambda_i a_{i1})x_1 + \cdots + (\sum_{i=1}^{m} \lambda_i a_{in})x_n \geq \lceil \sum_{i=1}^{m} \lambda_i b_i \rceil}$$

where $m \leq n$, the λ_i's are non-negative rational coefficients satisfying $\sum_{i=1}^{m} \lambda_i a_{ij} \in \mathbb{Z}$ for all $1 \leq j \leq n$, every $a_{ij} \in \mathbb{Z}$ and every $b_i \in \mathbb{R}$.

We translate the clauses of the original CNF formula into inequalities as follows: the clause

$$x_{i_1} \vee \cdots \vee x_{i_t} \vee \neg x_{j_1} \vee \cdots \vee \neg x_{j_f}$$

becomes the inequality

$$x_{i_1} + \cdots + x_{i_t} + (1 - x_{j_1}) + \cdots + (1 - x_{j_f}) \geq 1.$$

The rank of a polytope is the minimum number of rounds of applications of the cut rule required to reach its integer hull. If the converted CNF is contradictory as a linear program, then its CP rank is 0; if one round of the cut rule is enough to reach the integer hull of the polytope then its CP rank is 1. In general if i rounds of cuts are sufficient to reach a contradiction but $i-1$ rounds is not, then the rank of the polytope is i. We refer to the polytope defined by the converted CNF as P^0 and the polytope containing only points that can not be removed from P^0 in using i rounds of cuts as P^i. The integer hull of P^0 we call P_I. As a proof system, CP is both sound, since integral points can not be removed using the cut rule, and complete. The completeness of CP follows from a result of [13], which states that for bounded polyhedra there exists a j, such that $P^j = P_I$, here j is the rank of the polytope P. It also follows from the fact that it can easily be shown that CP can simulate Resolution ([18]).

Note that, as in [12], we can view the polytope resulting from a single round of applications of the cut rule, as follows:

$$P' = \{x \in P : \langle a, x\rangle \geq \lceil b \rceil \text{ whenever } a \in \mathbb{Z}, b \in \mathbb{R}, \text{ and } \langle a, y\rangle \geq b \text{ for all } y \in P\}.$$

Here we take $\langle\rangle$ to mean the standard inner-product. With this definition we have that $P^1 = (P^0)'$ and in general $P^{i+1} = (P^i)'$.

We define PHP by a collection of two sets of linear inequalities, which we will call the *holeset* and the *pigeonset*. For convenience we use variables numbered with ordered pairs; the variable $P_{(i,j)}$ represents the proposition "pigeon i goes into hole j". The *holeset* ensures that no two pigeons are assigned to the same hole and consists of all inequalities of the form $P_{(i,j)} + P_{(i',j)} \leq 1$, where $i \neq i'$, $1 \leq i \leq m$, $1 \leq i' \leq m$ and $1 \leq j \leq n$ for some given number of holes n and some number of pigeons m where $m > n$. The *pigeonset* states that each pigeon must go to at least one hole; this set consists of all inequalities of the form $\sum_{j=1}^{n} P_{(i,j)} \geq 1$, for every $1 \leq i \leq m$. Throughout this chapter we refer to the polytope defined by the *holeset* and *pigeonset* inequalities as PHP^0 or simply PHP and the polytope remaining after i rounds of applications of the cut rule as PHP^i. From now on the values of m and n are implicit.

4.3 Results

We first present a short proof that PHP^0 has a CP proof of rank $O(\log n)$ and of polynomial size (i.e. in the number of inequalities in the proof). Although, as previously mentioned, this rank upper bound has already been shown in previous proofs, these proofs were all of size $\Omega(n^{\log n})$.

Theorem 4.3.1 PHP^0 has a CP proof of rank $O(\log(n))$ and of polynomial size.

Proof: Suppose there are three disjoint sets of indices P,Q and R and that we have the inequalities $\sum_{p\in P} a_p + \sum_{q\in Q} a_q \leq 1$, $\sum_{p\in P} a_p + \sum_{r\in R} a_r \leq 1$ and $\sum_{q\in Q} a_q + \sum_{r\in R} a_r \leq 1$. We can generate the inequality $\sum_{p\in P} a_p + \sum_{q\in Q} a_q + \sum_{r\in R} a_r \leq 1$ in a single cut; by adding the three inequalities together, dividing the result by two and rounding down. Note that if the resulting inequality has k variables then it can be that we have P, Q and R such that $|P+Q|, |P+R|, |Q+R| \leq \lceil 2k/3 \rceil$.

Since for a given hole j we are given all inequalities of the form $P_{i,j} + P_{i',j} \leq 1$, where $1 \leq i, i' \leq m$ and $i \neq i'$, we know that, by the above argument, we are able to generate the inequality $\sum_{i=1}^{n+1} P_{i,j} \leq 1$. The rank of this inequality can be described by the recurrence $\Gamma(n) \leq 1 + \Gamma(\lceil 2n/3 \rceil)$, which yields a rank bound of $O(\log_{3/2} n)$. The number of cuts required to create it is described by the recurrence $\Delta(n) \leq 1 + 3\Delta(\lceil 2n/3 \rceil)$. This recurrence yields an upper bound of $O(3^{log_{3/2}(n)})$, which in turn can be rewritten as $O(n^{1/(1-\log_3(2))})$.

To reach a contradiction, we generate the inequality $\sum_{i=1}^{m} P_{i,j} \leq 1$ for all $1 \leq j \leq n$ then sum these inequalities together with all the *pigeonset* inequalities which yields the contradiction $m \leq n$. It is clear that this proof is of rank $O(\log n)$ and that the size of the proof is polynomial in n. □

In order to produce our rank lower bound, we first introduce a lemma which allows us to ensure certain points within the polytope survive a single round of cuts, under the condition that other points are also present in the polytope. Such lemmas are known as *protection* lemmas, since they demonstrate that a point is protected from being cut.

We begin with some notation. Let P be a polytope in the n dimensional 0-1 hypercube (i.e. $P \subseteq [0,1]^n$). We call a point x a *good point* for P if it has the following properties:

1. $x \in P$.
2. Each non-integral coordinate x_i of x has the value z_i/k where $z_i \in \mathbb{Z}$ for some fixed $k \in \mathbb{R}$.
3. Each coordinate x_i of x is less than or equal to $1/2$ unless $x_i = 1$.

Let a be a vector of integers of length n (i.e. $a \in \mathbb{Z}^n$) and b be a real value such that $\langle a, y\rangle \geq b$ for all points $y \in P$. We call such a pair (a, b) a *satisfying pair* for P. Let x be a good point of P, with each such a and x we associate a set of indices $J = \{j : a_j \neq 0 \text{ and } x_j \notin \mathbb{Z}\}$.

Note that if $\sum_{j\in J} |a_j x_j| \geq 1$, then there exists a $J^* \subseteq J$, such that $\sum_{j\in J^*} |a_j x_j| \geq 1$ and $|J^*| \leq k$. This is significant because the existence or non-existence of such a set J^* determines how we generate the point which we use to show x survives a given cut.

If such a set J^* exists, which happens when $\sum_{j\in J} |a_j x_j| \geq 1$, then we can associate with it a point t which is constructed by setting its coordinates as follows:

- $t_i = x_i$ for every $1 \leq i \leq n$ where $i \notin J^*$.
- $t_i = 0$ for every $1 \leq i \leq n$ where $i \in J^*$ where $a_i > 0$.
- $t_i = 2x_i$ for every $1 \leq i \leq n$ where $i \in J^*$ where $a_i < 0$

We call such a point a t-point of x and a. We also say that the t-point was created using the set J^*.

Proposition 4.3.1 Suppose there exists a set J^* for a given good point x and a satisfying pair (a, b) for some polytope $P \subseteq [0,1]^n$. If the t-point, t, created using J^* is in P, then $\langle a, x\rangle \geq \lceil b \rceil$.

Proof: From the way in which t is constructed it must be that $a_j t_j + |a_j x_j| = a_j x_j$ for each $j \in J^*$. Summing over all such equalities allows us to derive that $\langle a, t\rangle + 1 \leq \langle a, x\rangle$. Since we also know $\langle a, t\rangle \geq b$, as $t \in P$, we can be sure that $\langle a, x\rangle \geq \langle a, t\rangle + 1 \geq b + 1 \geq \lceil b \rceil$. □

If no J^* exists for a particular good point x and satisfying pair (a, b) for a polytope P, we construct another point s by setting its coordinates as follows:

- $s_i = x_i$ whenever $x_i \in \mathbb{Z}$.
- $s_i \in \{0, x_i\}$ for every $1 \leq i \leq n$ where $a_i = 0$ and $x_i \notin \mathbb{Z}$.
- $s_i = 0$ for every $1 \leq i \leq n$ where $a_i > 0$ and $x_i \notin \mathbb{Z}$.
- Let $G \subseteq J$ such that $g \in G$ if and only if $a_g < 0$. If $|G| \geq 1$, then s has a single coordinate $s_f = 1$ where $f \in G$ and has $s_{f'} = 0$ for all $f' \in G$ where $f' \neq f$.

We call a point created in such a fashion an s-point of x and a.

Proposition 4.3.2 Suppose no such set J^* exists for a given good point x and a satisfying pair (a, b) for some polytope $P \subseteq [0, 1]^n$. If there exists an s-point, s, of x and a such that $s \in P$, then $\langle a, x\rangle \geq \lceil b \rceil$

Proof: Since s_i is integral for each non-zero $a_i \in a$, we know that $\langle a, s\rangle \in \mathbb{Z}$. As $s \in P$, it must be that $\langle a, s\rangle \geq b$, and hence it must also be that $\langle a, s\rangle \geq \lceil b \rceil$. To see that $\langle a, s\rangle \leq \langle a, x\rangle$, from which the result follows, note that $a_i x_i > a_i s_i$ for each $j \in J$ where $a_j > 0$ and that if $|G| \geq 1$, then $\sum_{g \in G} a_g s_g \leq -1 \leq \sum_{g \in G} a_g x_g$. The first part of this inequality (i.e. $\sum_{g \in G} a_g s_g \leq -1$) holds because for a single $g \in G$, $s_g = 1$ and a_g is a negative integer smaller than 0, whilst for all other $g' \in G$, where $g' \neq g$ have $s_{g'} = 0$. The second part of the inequality ($-1 \leq \sum_{g \in G} a_g x_g$) holds because no J^* exists. □

Lemma 4.3.2 (**Protection**) Let x be a good point of a polytope $P \subseteq [0, 1]^n$. If for all possible satisfying pairs (a, b) of P, a t-point or an s-point of x and a is in P, then $x \in P'$.

Proof: It is clear that under these conditions, by Propositions 4.3.1 and 4.3.2, x must satisfy all the defining inequalities of P'. □

To help us describe how we use Lemma 4.3.2 to produce the rank lower bound for PHP, it is convenient to introduce some notation. For the rest of this chapter, we consider E_j to be the set of variables $P_{(i,j)}$ for all $1 \leq i \leq m$ and we say E_j is fixed on $P_{(i,j)}$ if every variable in E_j is set to 0, except $P_{(i,j)}$ which is set to 1; if this is not the case we say that E_j is unfixed. We define W_q to be the set of all points w on the variables of PHP that satisfy the following conditions:

1. Every coordinate of w is a non-negative multiple of $1/\sqrt{n}$.
2. At most $q\sqrt{n}$ sets E_j where $1 \leq j \leq n$ contain a variable with the value 0.
3. No coordinate of w is greater than $2^q/\sqrt{n}$ in any unfixed E_j.

Lemma 4.3.3 Every point $w \in W_q$, where $1 \leq q \leq \log(n)/4$, is a good point of PHP^0.

Proof: Each w clearly satisfies conditions 2 and 3 of the definition of a good point of a polytope (where $k = \sqrt{n}$), so we will focus on showing that $w \in \text{PHP}^0$, from which the result follows.

Each $w \in W_q$ satisfies the *pigeonset* inequalities ($\sum_{j=1}^{n} P_{(i,j)} \geq 1$ for all $1 \leq i \leq m$) since each such inequality must have at least $n - q\sqrt{n}$ variables set to at least $1/\sqrt{n}$, therefore when even when q is as large as possible we have:

$$\begin{aligned}(n - q\sqrt{n})\frac{1}{\sqrt{n}} &\geq 1 \\ (n - \frac{\log(n)}{4}\sqrt{n})\frac{1}{\sqrt{n}} &\geq 1 \\ \sqrt{n} - \frac{\log(n)}{4} &\geq 1.\end{aligned}$$

Each w can also be shown to satisfy the *holeset* inequalities. It trivially satisfies any such inequality $P_{(i,j)} + P_{(i',j)} \leq 1$ for any j where E_j is fixed. If E_j is unfixed, then as $q \leq \log(n)/4$, we have that $P_{(i,j)}, P_{(i'j)} \leq 1/2$ in w for sufficiently large n. This can be shown as follows:

$$P_{(i,j)}, P_{(i',j)} \leq 2^{\log(n)/4}/\sqrt{n} = (2^{\log(n)})^{1/4}/\sqrt{n} = n^{1/4}/n^{1/2} = 1/n^{1/4} \leq 1/2.$$ □

Lemma 4.3.4 Let $w \in W_q$, where $q \leq \log(n)/4 - 1$. We can find a set Q of s-points and t-points of w, satisfying Lemma 4.3.2 (where x is w), with each such point being in W_{q+1}.

Proof: Consider a satisfying pair (a, b) for PHP. If there exists a set J^* for this particular w and a, then the t-point, t, created using this J^* has no value greater than $2^{q+1}/\sqrt{n}$, and no more than $(q+1)\sqrt{n}$ sets E_j contain a variable set to zero; therefore, $t \in W_{q+1}$.

If no such J^* exists for w and a, then there can only be at most $\sqrt{n}$ elements in the set J. We create a point s from w as follows: if J contains at least one element (i, j) where $a_{(i,j)} < 0$, then we fix E_j on $P_{(i,j)}$ in s, for all remaining $(i', j') \in J$ where $1 \leq i' \leq m$, $1 \leq j' \leq n$ and $j' \neq j$, $s_{(i',j')} = 0$. All remaining coordinates of s (i.e. those not in J) are set to the same value as in w. Note that s is a s-point of w and a and that s has at most $(q+1)\sqrt{n}$ sets E_j that contains a variable set to zero. Furthermore s does not contain any value greater than $2^{q+1}/\sqrt{n}$ in any unfixed E_j and hence $s \in W_{q+1}$.

For each satisfying pair valid for PHP we have that either s or t is in Q. Since each such point in is W_{q+1}, by Lemma 4.3.3, they must also satisfy PHP, hence the conditions of Lemma 4.3.2 are met for w by Q. □

From now on we will refer to such a set Q, as in Lemma 4.3.4, as a *protective set* of w.

Theorem 4.3.5 The Chvátal rank of PHP is $\Omega(\log n)$.

Proof: It is enough to show that $\text{PHP}^{\log(n)/4}$ is non-empty. We demonstrate that this is the case by showing it must contain the point $x = [1/\sqrt{n}]^{m \times n}$ (i.e. x is the $m \times n$ dimensional point with each coordinate being $1/\sqrt{n}$). It is easy to see that $x \in W_0$ and by Lemma 4.3.4, we know x has a protective set of points Q_1, where every $q \in Q_1$, is in W_1 and hence by Lemma 4.3.3, $q \in \text{PHP}^0$. By Lemma 4.3.2, the existence of Q_1 demonstrates that $x \in \text{PHP}^1$. Now let q be a point in Q_1, we know by Lemma 4.3.4 that there is a protective set for q, consisting only of points in W_2; again by Lemmas 4.3.3 and 4.3.2 this demonstrates that $q \in \text{PHP}^1$. Such a protective set exists for all $q \in Q_1$; let Q_2 denote the union of all such sets. The

existence of Q_2, means that, again by Lemma 4.3.2, $x \in \text{PHP}^2$. Similarly we can define the set Q_p to be the union of the protective sets for every $q \in Q_{p-1}$. For convenience we define Q_0 to be the point x.

We can now complete the proof by induction. Assume that for some $p \leq \log(n)/4 - 1$ we have that every point in Q_r is also in W_r for all $0 \leq r \leq p$. We have already shown that this assumption holds for $p = 2$. Let q be a point in Q_p, we know from Lemma 4.3.4 there exists a protective set for q, consisting of points in W_{q+1} and hence by Lemma 4.3.3, also in PHP^0. Since such a protective set can be found for all such $q \in Q_p$, we know that all points in Q_p are in PHP^1. In general if all points in Q_d are in $\text{PHP}^{p-(d-1)}$ for some $1 \leq d \leq p$ then by Lemma 4.3.2 every point in Q_{d-1} is in $\text{PHP}^{p-(d-2)}$. Therefore it must be that $x \in \text{PHP}^{log(n)/4}$. □

Theorems 4.3.1 and 4.3.5 allow us to conclude the following corollary:

Corollary 4.3.6 The Chvátal rank of PHP is $\Theta(\log n)$.

4.4 Further Work

An interesting problem, following directly from this work, is to see whether the technique presented here can be used to prove a rank complexity gap, similar to those given in [22], for CP. We conjecture that all polytopes derived from sentences of first-order logic (as in [61]) possessing an infinite, but no finite model, require strictly non-constant CP rank, whilst all polytopes derived from sentences with no finite nor infinite model require just constant CP rank.

Chapter 5

Comparing the Rank Complexity of LS_+ and Cutting Planes to Resolution Width

5.1 Introduction

It is not difficult to show that all the proof systems defined by Lovász and Schrijver (LS_0, LS and LS_+), the Sherali Adams proof system (SA) and the Cutting Planes proof system (CP) can all p-simulate Resolution. However, whilst it is known that SA has the property that the rank of any polytope derived from a set of unsatisifiable clauses is at most the Resolution width of the clauses (Corollary 3.2.1), it is not clear whether this holds for any of the other abovementioned systems.

In this chapter we show that this is not the case; the measures of CP and LS_+ (the most powerful of the systems defined by Lovász and Schrijver) rank are incomparable to Resolution width. This result allows us to obtain a number of corollaries, which describe how the measures of rank in the various systems are related to each other.

5.2 Related Work

The Pigeonhole Principle provides an example of an infinite family of unsatisfiable CNF formuale which requires linear Resolution width ([36]) but only logarithmic

CP rank ([12]) and rank of just one in LS_+ ([35]). However it is unknown whether an upper bound for the Resolution width of an unsatisfiable CNF formulae implies an upper bound on the CP or LS_+ rank of its corresponding polytope. We show that this is not true by demonstrating that there is an infinite family of tautologies which require constant Resolution width, yet at least logarithmic CP and LS_+ rank.

It is shown in [48] that the SA rank of a polytope is no greater than its rank in the standard system devised by Lovász and Schrijver (LS), however there are no known examples which separate the two complexity measures. In this chapter we provide such an example, demonstrating the SA rank of a polytope can be arbitrarily smaller than its LS rank.

5.3 Preliminaries

In this chapter we describe upper and lower bounds on the complexity of proofs in terms of Resolution width and CP and LS rank; these are defined in Sections 2.2 and 2.3 respectively.

5.3.1 Relativized House Sitting Principle

The infinite family of unsatisfiable CNF formulae that we consider in this chapter we will call $RHSP2_n$, which stands for the Relativized House Sitting Principle with 2 sets. This is the family of CNF formulae generated by taking the normal House Sitting Principle (HSP) as defined in Section 2.4.4, and which can be defined in first-order logic as the constraints: $\forall x \exists y((y \geq x) \wedge W(x,y))$, $\neg W(0,0)$, $\forall x, y((x < y) \wedge W(y,y) \rightarrow \neg W(x,y))$, relativizing it twice (see [21]) and converting it to a purely propositional sentence over n possible witnesses ([61]). The reason we consider the relativized version of the HSP is that the original version has CP and LS rank 0, so would not be appropriate. The reason we consider the version with two sets instead of one is simply that the method we employ to obtain our rank lower bounds fails on the later version.

The formula $RHSP2_n$ can informally be considered to represent the contradictory scenario where there is a street with n houses on it, and the higher the house

number, the better it is. House one is a run-down shack and house n is a luxurious mansion. The residents living on the street decide to play a game in which they can go to each others houses, under a number of conditions. Since no-one wants a bad deal there agree only to go into a house at least as good as their own. The residents of house one obviously don't like their own house and so don't want to stay there. They also belong to two groups, the neighborhood watch (q) and a risky pyramid scheme (r). The residents of the street decide that if people who are in both these groups go into their house, they should join these groups as well. They also decide that if the owners of a house are in both these groups and stay their own homes, then no-one belonging to both the groups is allowed to visit their house. We represent the proposition "some of the owners of house i go to house j" with the variable $W_{i,j}$ and the proposition "the owners of house i belong to q (r)" using the variable S_i^q (S_i^r). We consider the formula $RHSP2_n$, where $n \geq 2$, as being defined by the following inequalities (clauses):

$\sum_{j=i}^{n} W_{i,j} \geq 1$ ($\bigvee_{j=i}^{n} W_{i,j}$), for all $1 \leq i \leq n$, which we shall call the *witnessing* inequalities and can be considered to state that the residents of house i must go to a house at least as good as there own.

$2 + S_j^t \geq W_{i,j} + S_i^q + S_i^r$ ($S_j^t \vee \neg W_{i,j} \vee \neg S_i^q \vee \neg S_i^r$) for all $t \in \{q, r\}$, $i \leq n-1$ and all $j \geq i+1$, $j \leq n$. We refer to this set of inequalities as the *inductive* ones. These state that if residents of house i are in both groups and they go to house j, then the residents of house j must be in the set t.

$W_{i,j} + W_{j,j} + S_i^q + S_i^r + S_j^q + S_j^r \leq 5$ ($\neg W_{i,j} \vee \neg W_{j,j} \vee \neg S_i^q \vee \neg S_i^r \vee \neg S_j^q \vee \neg S_j^r$), for all $1 \leq i \leq n-1$, $2 \leq j \leq n$ which we will refer to as the $fullhouse$ inequalities. These state that if the residents of houses i and j are in both groups, and j goes into their own house then i can't go to j's house.

A set of single clause inequalities, $W_{1,1} \leq 0$ ($\neg W_{1,1}$) which states that the residents of house one don't stay in their own house, $S_1^q \geq 1$ (S_1^q) and $S_1^r \geq 1$ (S_1^r),

which state that residents of house one are in both sets.

5.4 Results

To get our constant upper bound on the narrow Resolution width of $RHSP2_n$ we use the following witnessing pebbling game, introduced in [31].

Let F be a CNF formula. The witnessing pebble game on F is played between two players Prover and Delayer on the set of literals arising from the variables in F. A pebble can never appear on both the positive and negative literals of any variable. In each round, one of three things can happen.

1. Prover lifts a pebble from the board; Delayer makes no response.

2. (Querying a Variable.) Prover gives a pebble to Delayer and names an empty variable x (i.e. neither x nor $\neg x$ is pebbled already). Delayer then places the pebble on x or $\neg x$.

3. (Querying a Clause.) Prover gives Delayer a pebble and names a clause C from F. Delayer must then place the pebble on one of the literals of C if none are already pebbled, without contradicting a pebble already on the board, if at least one literal of C is already pebbled they can hand the pebble back to Prover. If this is impossible to do Prover wins the game.

When the game is limited to a given number of pebbles k, we call this the k-pebble witnessing game. Notice that Prover can only win if the pebbles on the board falsify a clause of F and Prover has at least one pebble left. From [31], we also get the following lemma, linking the witnessing pebble game with the narrow Resolution width of a proof.

Lemma 5.4.1 (Proposition 4, [31]) Let F be a CNF formula. If there is a winning

strategy for Prover in the k-pebble witnessing game on F, then there is a narrow Resolution proof of width k of the unsatisfiability of F.

We can now prove our constant upper bound on the Resolution width of $RHSP2_n$, by demonstrating that the value of n does not affect the number of pebbles required for Prover to win the witnessing pebble game on the clauses of $RHSP2_n$.

Theorem 5.4.2 For every $n \geq 3$, $RHSP2_n$ has a narrow Resolution proof of width ≤ 6.

Proof: At the start of the game Prover queries the single literal clauses S_1^q, S_1^r and $\neg W_{1,1}$.

We will now show that if there are pebbles on S_i^q, S_i^r, $\neg W_{i,i}$ for some value i, then Prover can force Delayer to placed pebbles on S_j^q, S_j^r, $\neg W_{j,j}$ for some $j > i$ using just 3 more pebbles. To do this Prover first queries the *witnessing* clause $\bigvee_{q=i}^{n} W_{i,q}$, Delayer must put a pebble on some literal $W_{i,j}$. Prover then queries the variables S_j^q and S_j^r, Delayer must play on the positive literals of both of these variables or Prover could then query the one of the *inductive* clauses $S_j^t \vee \neg W_{i,j} \vee \neg S_i^q \vee \neg S_i^r$ for each $t \in \{q, r\}$ and win straight away. Prover then queries the $fullhouse$ clause $\neg W_{j,j} \vee \neg S_i^q \vee \neg S_i^r \vee \neg S_j^q \vee \neg S_j^r$; in response, Delayer must place a pebble on $\neg W_{j,j}$.

Note that if Prover plays in this manner, having secured pebbles on S_j^q, S_j^r and $\neg W_{j,j}$, they can then pick up the pebbles on S_i^q, S_i^r and $W_{i,j}$ and repeat the same strategy. If Prover continually plays in this manner, then eventually they can force Prover to pebble S_n^q, S_n^r and $\neg W_{n,n}$; they can then win by querying the single literal clause $W_{n,n}$. By following this strategy, Prover is able to win using just six pebbles. □

To prove our logarithmic rank lower bound for LS_+ and CP on RHSP_2^n, we need some lemmas, which we refer to as *protection* lemmas, which ensure some specific point can not be cut from the current polytope in the next round of applications of the respective cut rule, under the condition that other points are also present in the current polytope.

For CP we use the following protection lemma, as presented in [12].

Lemma 5.4.3 (Lemma 3.1, [12]) Let P be a bounded polytope in $\mathbb{R}^n$ and let P' be the polytope defined by applying a round of applications of the cut-rule to P. Let $y \in P$ be a point which has each of its coordinates in the set $\{0, \frac{1}{2}, 1\}$ and let $V = V(y)$ be the set of coordinates for which y is $\frac{1}{2}$. Let V be partitioned into distinct sets $V_1, V_2, \ldots, V_t$. Suppose that for every $j \in \{1, 2, \ldots, t\}$ we can represent y as an average of two vectors both in P that are 0/1 on V_j and agree with y elsewhere. Then $y \in P'$.

We now show that Lemma 5.4.3 also holds where P' refers to the polytope defined similarly except that instead of applying the cut-rule we apply N_+-cuts. In order to do this we first need the following lemma from [25]; note that we use the notation presented previously in Section 2.3.2.

Lemma 5.4.4 (Lemma 3.2, [25]) Let P' be the polytope remaining after applying a round of N_+-cuts to a given polytope P. A point x is in P' if there exists an $(n+1) \times (n+1)$ PSD matrix Y, such that $Y(e_0) = x$, $diag(Y) = x$, for each $0 \leq i \leq n$, $Y(e_i) \in P$ and $Y(e_0) - Y(e_i) \in P$.

Using Lemma 5.4.4, we can show that under the conditions of Lemma 5.4.3, the given point would also survive a round of N_+-cuts.

Lemma 5.4.5 Let P be a bounded polytope in $\mathbb{R}^n$ and let P' be the polytope remaining after applying a round of N_+-cuts to P. Let $y \in P$ be a point which has each of its coordinates in the set $\{0, \frac{1}{2}, 1\}$ and let $V = V(y)$ be the set of coordinates for which y is $\frac{1}{2}$. Let V be partitioned into distinct sets $V_1, V_2, \ldots, V_t$. Suppose that for every $j \in \{1, 2, \ldots, t\}$ we can represent y as an average of two vectors both in P that are 0/1 on V_j and agree with y elsewhere. Then $y \in P'$.

Proof: We demonstrate this by showing that under the conditions of this lemma, one can build a PSD matrix, satisfying the conditions of Lemma 5.4.4 for the point y.

Consider the matrix E constructed as having $E_{0,0} = 1$, $diag(E) = y$, $E_0 = y$ and the zero-th row of E is also y (i.e. $E_{0,i} = y_i$ for each $i \in \{1, \ldots n\}$). For every i where $y_i = 0$ we have that E_i is the all zero vector and the i-th row of E is the all

zero row vector. If $y_i = 1$ then we have that $E_i = y$ and the i-th row of E is also y. If $y_i = \frac{1}{2}$, and $i \in V_c$ then if $y_j = \frac{1}{2}$ and $j \notin V_c$ then $E_{i,j}, E_{j,i} = \frac{1}{4}$. In the case that $i, j \in V_c$ and $y_i = y_j$ in the two points that are 0/1 on V_c and average to y, then $E_{i,j}, E_{j,i} = \frac{1}{2}$ and if $y_j \neq y_i$ in these two points then $E_{i,j}, E_{j,i} = 0$.

For example if $y = [\frac{1}{2}, \frac{1}{2}, 1, 0, \frac{1}{2}, \frac{1}{2}]$ and we have $V_1 = \{1, 2\}$, $V_2 = \{5, 6\}$, y_1 and y_2 are set to the same value as each other in the points that are 0/1 on V_1, and that y_5 and y_6 are set to the opposite value to each other in the points that are 0/1 on V_2, then

$$E = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} & 1 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} \\ 1 & \frac{1}{2} & \frac{1}{2} & 1 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{pmatrix}.$$

If y_i is integral, then we have that one of E_i, $E_0 - E_i$ is the all zero vector of length $n + 1$ and the other is the vector defines the point y, therefore both these points must be in P. Note that if $y_i = \frac{1}{2}$ and $i \in V_c$, then E_i and $E_0 - E_i$ define points which are 0/1 on V_c and exactly the same as y on each coordinate not in V_c. From the way in which we constructed E, these two points are precisely those which we know to be in P and average to y.

Since E is clearly always symmetric, if we were showing that Lemma 5.4.4 holds for LS we would be finished. To finish the proof for LS_+ we need only check that the matrix E is always PSD. Consider the matrix $A = y \times y^T$, this matrix is clearly PSD and is very similar to E. A differs from E only on diagonal entries $E_{i,i}$ where $y_i = \frac{1}{2}$, in which case we have that $A_{i,i} + \frac{1}{4} = E_{i,i}$ and also on entries $E_{i,j}$ (and by symmetry $E_{j,i}$) where $y_i, y_j = \frac{1}{2}$ and $\{i, j\} \subseteq V_c$, where it differs by either plus or

minus $\frac{1}{4}$. For instance, in the example given before, we have that

$$A = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} & 1 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} \\ 1 & \frac{1}{2} & \frac{1}{2} & 1 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} \end{pmatrix}.$$

Let q be the smallest index in V_i and let B_i be an $(n+1) \times (n+1)$ matrix containing all zeros except that $E_{j,1} = \frac{1}{2}$ if and only if $j \in V_i$ and $v_q = v_j$ in the two points that are 0/1 on V_i and the same as y elsewhere and $E_{j,1} = -\frac{1}{2}$ if and only if $j \in V_i$ and $v_q \neq v_j$ in these points. For instance in the example presented before:

$$B_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} B_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Now consider the matrix $B_i \times B_i^T$. This is clearly a PSD matrix of dimension $(n+1) \times (n+1)$ and has entries $E_{j,j} = \frac{1}{4}$ for every $j \in V_i$. It also has the value $\frac{1}{4}$ for every entry $E_{j,t}$ ($E_{t,j}$) where $\{j,t\} \subseteq V_i$ and $y_j = y_t$ in the two points that are 0/1 on V_i and the value of y elsewhere. However, if $y_j \neq y_t$ in these points then $E_{j,t}, E_{t,j} = -\frac{1}{4}$. All other entries in $B_i \times B_i^T$ are zeros. Given this, one can see that $A + \sum_{i=1}^{t} B_i \times B_i^T = E$. Since each $B_i \times B_i^T$ is PSD and A is PSD, it must be that E is PSD too. □

For the rest of this chapter, we consider P' to represent the polytope remaining after applying a round of applications of both the cut-rule and N_+-cuts to a given polytope P, that P^0 is the polytope defined by the inequalities of $RHSP2_n$, and that for each i, $P^{i+1} = (P^i)'$. The proof of our lower bound on the CP and LS_+ rank of $RHSP2_n$ involves showing that a specific point x, defined as having the variables set to the following values, must be present in $P^{\lfloor \log_2(n) \rfloor - 2}$.

$W_{n,n}$, S_1^q, $S_1^r = 1$.
$W_{i,i+1}$, $W_{i,i+2} = \frac{1}{2}$ for all odd i, where $1 \leq i \leq n-3$.
$W_{i,i+2}$, $W_{i,i+3} = \frac{1}{2}$ for all even i where $2 \leq i \leq n-4$.
S_i^q, $S_i^r = \frac{1}{2}$ for all $2 \leq i \leq n$.
$W_{n-2,n-2}$, $W_{n-2,n-1}$, $W_{n-1,n-1}$, $W_{n-1,n} = \frac{1}{2}$.
All other variables are set to 0.

Note that by substituting the values given in x, if n is large enough, each inequality of $RHSP2_n$ is satisfied and hence $x \in P^0$.

Theorem 5.4.6 The CP and LS_+ rank of $RHSP2_n > \lfloor \log_2(n) \rfloor - 2$, where n is even and $n \geq 8$.

Proof: We demonstrate that $P^{\lfloor \log_2(n) \rfloor - 2}$ is non-empty by demonstrating that it contains x. For the rest of this proof we consider $V = V(y)$ to be the set of all coordinates for which the point y is non-integer.

We consider a game played between two players, Delayer and Prover, which runs over a number of rounds. In each round Prover is at some point y in the space (i.e. in the polytope P^0) and in the first round $y = x$. It is Prover's job to try and find a point $y \notin P^0$, it is Delayer's job to make the game last for as many rounds as possible. At the beginning of each round Delayer partitions the variables of V into distinct sets $V_1, \ldots, V_t$. For each set V_i of the partition of V Delayer decides on two possible assignments of 0/1 values to the variables in the set, where each variable is assigned 0 in one assignment and 1 in the other (i.e. as in lemmas 5.4.3 and 5.4.5). Prover then picks one of these sets and chooses one of the two possible assignments.

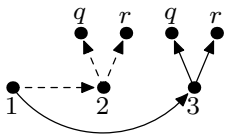


Figure 5.1: The set V_{start}, and associated 0/1 values.

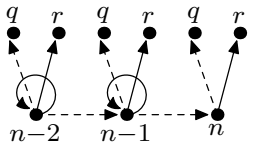


Figure 5.2: The set V_{end}, and associated 0/1 values.

They then update their position y by setting these variables to their chosen values. We call a position reached from a given position g in this manner a child of g.

Since the inequalities of $RHSP2_n$ are unsatisfiable, it is clear that Prover will eventually win the game (i.e. if he sets all the variables to 0/1 values he must have reached a point $y \notin P^0$). The clear link between the game and the CP and LS_+ rank is that if Delayer can play so that the game lasts until the end of round i, by Lemmas 5.4.3 and 5.4.5, the CP and LS_+ rank of the polytope P^0 is at least $i+1$. We therefore demonstrate a strategy for Delayer that allows him to ensure the game lasts until the end of round $\lfloor \log_2(n) \rfloor - 2$.

Figures 5.1 to 5.6, define the possible sets into which Delayer will partition $V = V(y)$ together with their associated 0/1 values which define the child points of a given current point. An edge uv in these figures represents the variable $W_{u,v}$ if v is not labeled q or r, otherwise it represents the variable S_u^v. The two possible child positions, having 0/1 values associated with a given set V_i of the partition of V, are defined as having all the dashed edges set to 0, and the solid edges 1 and vice-versa in the respective figure (and the same values as y for all variables not appearing in V_i).

Figures 5.1 and 5.2 show how specific elements of V are partitioned into subsets, however Figures 5.3 to 5.6 give a template defining how elements of V can be partitioned according to some parameter i. We call the set matching template T_a having $i = p$, $V_{a,p}$ and we say $V_{a,p}$ has 'i' value p.

Note that V_{start} and T_{left} are constructed so that if someone goes into some house

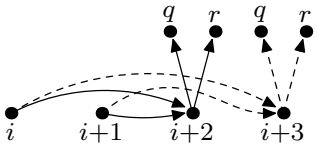


Figure 5.3: The template T_{left}, and associated 0/1 values.

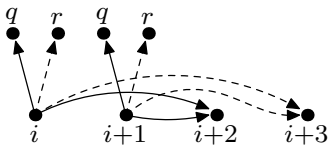


Figure 5.4: The template T_{right}, and associated 0/1 values.

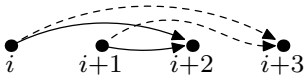


Figure 5.5: The template T_{mid}, and associated 0/1 values.

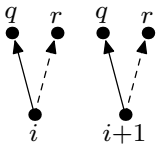


Figure 5.6: The template T_{other}, and associated 0/1 values.

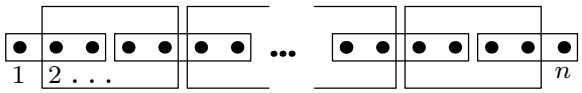


Figure 5.7: The board on which the game is played.

h, the residents of h are always in both q and r. By comparison, the other sets are constructed so as to prevent the residents of a particular house belonging to both q and r.

Intuition. There are distinct numbers, (i) the biggest number h so that the residents of house h are in both q and r and (i.e. $S_h^q = S_h^r = 1$) and (ii) the lowest number g so that the residents of house g are not definitely in both r and s (i.e. $S_g^q + S_g^r \neq 2$) and have some people staying in their house (i.e. $W_{k,g} = 1$ for at least one k in $\{1, \dots g\}$) or simply are not in one of q or r (i.e. $S_g^q = 0$ or $S_g^r = 0$). Only the segment between (i) and (ii) is inconsistent so Prover tries to narrow the size of this segment, whilst Delayer makes sure it can only be halved in each round, hence the $\Omega(\log_2 n)$ lower bound. A valid, although ineffective move for Prover is to play in the consistent region to the left of (i), or in the consistent region to the right of (ii). Initially, at the start of the game, we have that $(i) = 1$ and $(ii) = n$.

Whilst the link between the game we introduced and the CP and LS_+ rank is clear, to simplify the explanation of the Delayer's strategy we will introduce a few slight changes to the way in which it is presented. In this alternative presentation, the game is played on a board consisting of boxes laid over vertices labeled from 1 up to n. The first box is over the vertices 1 to 3, the last box covers the vertices $n-2$ to n and there is a box for each even number between 2 and $n-4$ inclusive, each one covering the vertex with that number together with the next three. We consider each box to be numbered with the smallest vertex that it covers, (e.g. the box covering vertices 4,5,6 and 7 is box 4). This board is presented pictorially in Figure 5.7.

In this version of the game, each set V_i of the partition of V is represented by a box, at the start of each round Delayer labels each box with a letter, the letter

represents the template or set type represented by the box. Box 1, must always be labeled with an 'S' for 'V_{start}', the final box $(n-2)$ is always labeled 'E' for 'V_{end}'; the other boxes are labeled with one of 'L' for T_{left}, 'R' for T_{right} or 'M' for T_{mid}. The number of the box is also the 'i' value of the set it represents, for example if box 2 is labeled with an 'L', this is representative of the Delayer having dealt the set $V_{left,2}$ in the original game; for simplicity we consider the sets V_{start} and V_{end} to have 'i' values 1 and $n-2$ respectively.

After Delayer has labeled the boxes, Prover picks a box, which is representative of the them removing the corresponding set in the original game, they then update their position y in the same way as before, by setting the variables in the set they selected to 0/1 values, according to the restrictions set by the set type they pick. The box Prover selected is then shaded out and can no longer be picked by Prover, nor can the label be changed by Delayer in subsequent rounds. The game ends as before when the position y violates at least one of the clauses of $RHSP2_n$.

In each round Delayer's strategy is to ensure that the box labeled 'M' remains in the middle of the inconsistent region, to do this he labels the box with the nearest even value to $(i) + ((ii) - (i))/2$, 'M', if there are two such values, we assume he picks the smaller of the two, we call this value m. He then labels all unshaded boxes numbered less than the m with 'L', except for box one, which is always labeled with 'S'. All unshaded boxes numbered higher than m he labels 'R' with the exception of box $n-2$, which again is always labeled 'E'. The labeling he makes at the start of the game is graphically represented in Figure 5.8. It can easily be checked that this represents a valid partitioning of the variables of $V(x)$. Note that if Player selects a box marked 'L' or 'S' with a higher number that any shaded box labeled 'L' or 'S', they will increase the value of (i), whilst picking a set labeled 'M', 'R' or 'E' numbered lower than any shaded such box will decrease the value of (ii). Also note that if Prover picks any other box, it is equivalent to them playing in the consistent region, and hence is a poor move for them.

The only complication arises when Prover picks the set labeled 'M'. In this case, the value of (ii) decreases to the previous 'mid point' of the inconsistent region, m, but Delayer can not simply use their strategy in the next round as this would not

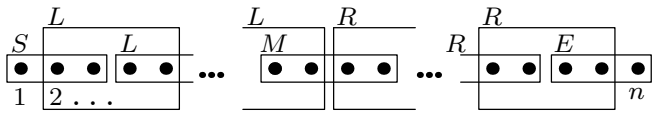


Figure 5.8: Delayer's deal in the first round, where each box is labeled above its top left hand corner.

constitute a valid partitioning of the variables of V. This is because the variables S^q_m, S^r_m, S^q_{m+1} and S^r_{m+1} would not be assigned to a set in the subsequent round. To cope with this, Delayer could simply deal these variables into the set $T_{other,m}$ in all subsequent rounds, but to simplify the strategy of Delayer, we assume that when Prover picks the box labeled 'M', they assign the variables associated with that box (i.e. $W_{m,m+2}$, $W_{m,m+3}$, $W_{m+1,m+2}$ and $W_{m+1,m+3}$) and then have a "free bonus round" in which they assign the variables of $T_{other,m}$ as if they had picked that set in the next round, although we don't count this as a round. This makes it equivalent to Prover having picked the box numbered m labeled with 'R'.

To prove that this strategy allows Delayer to play up until the end of round $\lfloor \log_2(n) \rfloor - 2$, we will show that it satisfies two properties, namely that firstly it always provides a valid partitioning of the set V into sets matching the templates or being V_{start} or V_{end} and secondly that y remains in P^0.

The first of these properties is easy to see. The strategy certainly is valid in round 1 (i.e. as in Figure 5.8), to see that it is valid in subsequent rounds note that Prover can only decrease the size of the inconsistent region (defined by $(ii) - (i)$) by at most half each time. Therefore so as long as the game only runs over at most $\lfloor \log_2(n) \rfloor - 2$ rounds, the inconsistent region will remain ≥ 8 on Delayer's last go. In terms of boxes, this means that Delayer always has at least 4 unshaded boxes numbered between (i) and (ii) and hence can ensure that there is at least one unshaded box to the left of the middle (i.e. the box labeled 'M') and one to the right. One can see that, taking into account the special rule about when Prover picks the unshaded set labeled 'M', such a labeling constitutes a valid partitioning of the set V.

To see that the strategy ensures the point y remains in P^0 we note that the only inconsistent situations are (1) when the residents of a house don't go anyway

(i.e. violating a *witnessing* inequality), (2) the residents of a house are in both sets and go to another house in which the residents are not in both sets (i.e. violating an *inductive* inequality) (3) the residents of a house in both sets occupy their own house and other people also in both sets go into that house (i.e. violating a $fullhouse$ inequality).

It is clear that (1) does not occur since each time Prover sets a variable $W_{i,j}$ to zero, they are forced to set another variable $W_{i,j'}$ to one. Situation (2) can only arise when the residents of a house are in both sets and go to another house ($S_i^q = S_i^r = W_{i,j} = 1$ for some i and j). This can only occur in y if Prover selects a set labeled either 'S' or 'L'. However, in this case no matter which of the two possible sets of values they assign to the variables in such a set, they are forced to make $S_j^q = S_j^r = 1$, thus preventing (2) from occurring. Situation (3) can only occur if the residents of a particular house stay in their own house, it is clear that this is only possible in y for the residents of houses $n-2$, $n-1$ and n. However, due to the limited number of rounds Delayer ensures that no matter how Prover plays the game they can not make (i) large enough for situation (3) to occur. Since none of (1), (2) or (3) can occur, we can be sure that $y \in P^0$ at the end of round $\lfloor \log_2(n) \rfloor - 2$ and hence our proof of Theorem 5.4.6 is complete. □

From Theorems 5.4.2 and 5.4.6 together with the logarithmic and constant Pigeonhole Principle rank bounds given in [12] and [35] for Cutting Planes and LS_+ respectively, and the size lower bound given in [36], we can conclude the following corollary.

Corollary 5.4.7 Narrow Resolution width is incomparable to CP and LS_+ rank.

As in the proof of Theorem 5.4.6 all variables $W_{i,j} = 0$ for all $j > i + 3$ in x, it is clear that we could get precisely the same CP and LS_+ rank lower bound for the polytope defined by all the inequalities in $RHSP2_n$, except having all these variables removed. This would give us an instance which had a maximum clause length of six, (i.e. a 6-CNF). In [31], they prove the following Lemma.

Lemma 5.4.8 (Proposition 2, [**31**]) If a *r-CNF*, F, has a width k narrow Resolution refutation, then F has a width $r + k - 2$ "normal" Resolution refutation.

Since the narrow Resolution width of a set of clauses can not increase if variables are removed, this lemma allows us to see that our altered version of $RHSP2_n$ has a "normal" Resolution refutation of width ten. This implies the following corollary.

Corollary 5.4.9 Resolution width, defined in terms of the maximum size of any clause in the proof, is incomparable to CP and LS_+ rank.

As the SA rank of the Pigeonhole Principle is known to linear in the number of pigeons [57], and the SA rank of any polytope derived from a set of unsatisfiable clauses is at most the normal Resolution width of the clauses (Corollary 3.2.1), we find that Theorems 5.4.6 and 5.4.2 also imply the following Corollary.

Corollary 5.4.10 SA rank is incomparable to CP and LS_+ rank.

Theorems 5.4.6 and 5.4.2 together with a result of [48], showing that SA can p-simulate LS such that the simulation is rank preserving, allow us to conclude the following corollary.

Corollary 5.4.11 The SA rank of a polytope can be arbitrarily smaller than its LS rank, but never larger.

5.5 Further Work

The results in the chapter fill in a gap in our knowledge about how CP, LS, LS_+ and SA rank and Resolution width are related. There are similar other gaps; for instance it is unknown whether CP and LS rank are incomparable or whether Resolution width and SA rank are equivalent.

One direct open question is whether $\Omega(\log n)$ is a tight lower bound on the CP or LS_+ rank of $RHSP2_n$, since it is possible that it could require linear rank.

Bibliography

[1] M. Ajtai. 1994. The complexity of the pigeonhole principle. *Combinatorica*, **14(4)**, 417-433.

[2] M. Alekhnovich, S. Arora and I. Tourlakis. 2005. Towards strong nonapproximability results in the Lovasz-Schrijver hierarchy. In *Proceedings of the thirty-seventh annual ACM Symposium on Theory of Computing, STOC '05*, 294-303.

[3] A. Atserias and M. Bonet. 2004. On the automatisability of resolution and related propositional proof systems. *Information and Computation archive*, **198(2)**, 182-201.

[4] P. Beam, R. Impagliazzo, J. Krajíček, T. Pitassi, P. Pudlak, A. Woods. Exponential lower bound to the size of bounded depth Frege proofs of the Pigeonhole Principle. In *Proceedings of the 24th annual ACM symposium on the Theory of Computing, STOC '92*, 200-220.

[5] P. Beame and T. Pitassi. 1996. Simplified and Improved Resolution Lower Bounds. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science, FOCS '96*, 274-282.

[6] P. Beame and T. Pitassi. 2001. Propositional Proof Complexity: Past, Present, and Future. In *Current Trends in Theoretical Computer Science: Entering the 21st Century*, World Scientific Publishing, 42-70.

[7] E. Ben-Sasson, A. Wigderson. 2001. Short proofs are narrow - resolution made simple. *Journal of the ACM*, **48(2)**, 149-168.

[8] D. Bienstock and N. Ozbay. 2003. Tree-width and the SheraliAdams operator. CORC REPORT 2003-09.

[9] A. Blake. 1937. Canonical expressions in boolean algebra, PhD. Thesis, University of Chicago.

[10] J.A. Bondy and U.S.R. Murty. 1976. Graph Theory with Applications. *North Holland, New York*.

[11] M.L. Bonet and M. Galesi. 1999. A Study of Proof Search Algorithms for Resolution and Polynomial Calculus. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, FOCS'99*, 422-432.

[12] J. Buresh-Oppenheim, N. Galesi, S. Hoory, A. Magen and T. Pitassi. 2006. Rank bounds and integrality gaps for cutting planes procedures. *Theory of Computing*, **2(2006)**, 65-90.

[13] V. Chvátal. 1973. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, **4**, 205-337.

[14] V. Chvátal and E. Szemereédi. 1988. Many hard examples for resolution. *Journal of the ACM*, **35(4)**, 759-768.

[15] S.A. Cook. 1971. The complexity of theorem proving methods. In *Proceedings of the 3rd annual ACM symposium on the Theory of Computing, STOC '71*, 151-158.

[16] S.A. Cook and R. Reckhow. March 1979. The relative complexity of propositional proof systems. *Journal of Symbolic Logic*, **44(1)** 36-50.

[17] S.A. Cook. 2000. The P vs NP problem. http://www.claymath.org/millennium/P_vs_NP/pvsnp.pdf.

[18] W. Cook, R. Coullard and G. Turan. 1987. On the complexity of cutting planes proofs. *Discrete Applied Mathematics*, **18**, 25-38.

[19] W. Cook and S. Dash. 1999. On the matrix-cut rank of polyhedra. *Mathematics of Operations Research*, **26(1)**, 19-30.

[20] T.H. Corman, C.E. Leiserson, R.L Rivest and C. Stein. 2001. Inrtoduction to Algorithms, Second Edition. The MIT Press and McGraw-Hill Book Company.

[21] S. Dantchev. 2006. Relativisation Provides Natural Separations for Resolution-Based Proof Systems. In *Proceeding of Computer Science - Theory and Applications, First International Computer Science Symposium in Russia, CSR '06*, 147-158.

[22] S. Dantchev. 2007. Rank complexity gap for Lovász-Schrijver and Sherali-Adams proof systems. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, STOC '07*, 311-317.

[23] S. Dantchev, B. Martin and M. Rhodes. 2008. Tight rank bounds for the Sherali-Adams proof system. Accepted to *Theoretical Computer Science*.

[24] G.B. Dantzig. 1991. Linear Programming. *History of Mathematical Programming: A Collection of Personal Reminiscences*, J. K. Lenstra, A. H. G. Rinnooy Kan, and A. Schrijver (eds.), Elsevier Science Publishers B.V., Amsterdam, The Netherlands.

[25] S. Dash. 2001. On the Matrix Cuts of Lovász and Schrijver and their use in Interger Programming. PhD. thesis. Computational and Applied Mathematics, Rice University, Houston, Texas.

[26] S. Dash. 2005. An exponential lower bound of the length of some classes of branch-and-cut proofs. *Mathematics of Operations Research*, **30(3)**, 678-700.

[27] M. Davis and H. Putnam. 1960. A computing procedure for quantification theory. *Communications of the ACM*, **7**, 201-215.

[28] M. Davis, G. Logemann and D. Loveland. 1962. A machie program for theorem proving. *Communications of the ACM*, **5**, 394-397.

[29] F. Eisenbrand and A.S. Schulz. 1999. Bounds on the Chvátal rank of polytopes in the 0/1-cube. In *IPCO'99*, *Lecture Notes in Computer Science*, **1610**, 137-150.

[30] A. Frieze and S. Suan. 1996. Analysis of two simple heuristics on a random instance of k-SAT. *Journal of Algorithms*, **20(2)**, 312-355.

[31] N. Galesi and N. Thapen. 2005. Resolution and Pebbling Games. In *Proceeding of SAT 2005*, *Lecture Notes in Computer Science*, **3569(2005)**, 76-90.

[32] M. R. Garey and D. S. Johnson. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences). W. H. Freeman.

[33] A. Goerdt. 1996. A threshold for unsatisfiablity. *Journal of Computer and System Sciences*, **53**, 469-486.

[34] R. Gomory. 1958. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the AMS*, **64**, 275-278.

[35] D. Grigoriev, E.A. Hirsch, and D. V. Pasechnik. 2002. Complexity of semi-algebraic proofs. *Moscow Mathematical Journal*, **4(2)**, 647-679.

[36] A. Haken. 1985. The intratability of resolution. *Theoretical Computer Science*, **39**, 297-308.

[37] M. Hartmann. 1988. Cutting Planes and the Complexity of the Integer Hull. PhD. Thesis. School of Operations Research and Industrial Engineering, College of Engineering, Cornell University.

[38] S. Hoory, N. Linial and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, **43(4)**, 439-561.

[39] R. A. Horn and C. R. Johnson. 1985. Matrix Analysis. Cambridge University Press.

[40] R. Impagliazzo, T. Pitassi and A. Urquhart. 1994. Upper and lower bounds on tree-like cutting plane proofs. In *Proceedings of the 9th IEEE Symposium on Logic in Computer Science, LICS '94*, 220-228.

[41] S. Jukna. 2001. Extremal Combinatorics with Applications in Computer Science. *Texts in Theoretical Computer Science*, Springer-Verlag, Berlin Heidelberg.

[42] A. Kamath, R. Motwani, K. Palem and P. Spirakis. 1995. Tail Bounds for Occupancy and the Satisfiability Threshold Conjecture. *Random Structures and Algorithms*, **7**, 59-80.

[43] R. M. Karp. 1972. Reducibility among combinatorial problems. *Complexity of Computer Computations*, New York: Plenum Press, New York, 85-103.

[44] L.G. Khachiyan. 1979. A polynomial time algorithm for linear programming. *Doklady Akademii Nauk SSSR*, *n.s.*, **244(5)**, 1063-1096. English translation in *Soviet Math. Dokl.* **20**, 191-194.

[45] J. Krajíček. 1994. Lower bounds to the size of constant-depth propositional proofs. *Journal of Symbolic Logic*, **39(1)**, 7386.

[46] J. Krajíček. 2001. On the weak pigeonhole principle. *Fundamenta Mathematicae*, **170**, 123-140.

[47] J. Krajíček. 2003. Lecture notes for Proof Complexity, TIN068, Spring 2005, Charles University.

[48] M. Laurent. 2003. A comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, **28(3)**, 470-496.

[49] L. Lovász and A. Schrijver. 1991. Cones of matrices and set functions and 0-1 optimization. *SIAM J. Optimization*, **7(1)**, 166-190.

[50] J. Marques-Silva. 2008. Practical applications of Boolean Satisfiability. In *Proceedings of the 9th International Workshop of Discrete Event Systems, WODES 2008*, 74-80.

[51] T. Pitassi, N. Segerlind. 2007. Exponential lower bounds and integrality gaps for tree-like Lovasz-Schrijver procedures. *Electronic Colloquium on Computational Complexity (ECCC)*, Tech. Rep. **TR07–107**.

[52] P. Pudlak. 1997. On the complexity of propostional calculas. In *Sets and Proofs*, Cambridge Press. Invited papers from Logic Colloquium.

[53] S.P. Radziszowski. 2006. Small Ramsey Numbers. *Electronic Journal of Combinatorics*, Dynamic Survey **1**, revision **11**.

[54] R. Raz. 2004. Resolution lower bounds for the weak pigeonhole principle. *Journal of the ACM*, **51(2)**, 115-138.

[55] A.A. Razborov. 2002. Resolution Lower Bounds for the Weak Functional Pigeonhole Principle. *Electronic Colloquium on Computational Complexity (ECCC)*, Tech. Rep. **TR01-021**.

[56] A.A. Razborov. 2002. Proof Complexity of Pigeonhole Principles. Developments in Language Theory. *Lecture Notes in Computer Science*, **2295**, 203-206.

[57] M. Rhodes. 2007. Rank lower bounds for the Sherali-Adams operator. In *Proceedings of CiE 2007, Lecture Notes in Computer Science*, **4497**, 648-659.

[58] M. Rhodes. 2008. On the Chvátal Rank of the Pigeonhole Principle. Accepted to *Theoretical Computer Science*.

[59] M. Rhodes. 2008. On the Rank of Proof Systems based on Integer Linear Programming. Submitted.

[60] M. Rhodes. 2008. Resolution Width and Cutting Plane Rank are Incomparable. In *Proceeding of MFCS 2008, Lecture Notes in Computer Science*, **5162**, 575-587.

[61] S. Riis. 2001. A complexity gap for tree-resolution. *Computational Complexity*, **10(3)**, 179-209.

[62] A. Schrijver. 1998. Theory of Linear and Integer Programming. Wiley.

[63] H.D. Sherali and W.P. Adams. 1990. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal of Discrete Mathematics*, **3**, 411-430.

[64] M. Sipser. 2005. Introduction to the Theory of Computation, second edition. Course Technology.

[65] G. Tseitin. 1968. On the complexity of derivation in the propositional calculas. *Studies in Constructive Mathematics and Mathematical Logic, Part II.*

[66] A. Urquhart. 1987. Hard Examples for Resolution. *Journal of the ACM*, **34(1)**, 209-219.

[67] J.P. Warners. 1998. A linear-time transformation of linear inequalities into conjunctive normal form. *Information Processing Letters*, **68**, 63-69.

[68] J.P. Warners. 1999. Nonlinear approaches to satisfiability problems. PhD thesis, Eindhoven University of Technology, The Netherlands.