Size Space tradeoffs for Resolution

Eli Ben-Sasson [*]
Electrical Engineering and Computer Science
Harvard University
Cambridge, MA
eli@eecs.harvard.edu

February 13, 2002

Abstract

We investigate tradeoffs of various important complexity measures such as *size*, *space* and *width*. We show examples of CNF formulas that have optimal proofs with respect to any one of these parameters, but optimizing one parameter must cost an increase in the other. These results, the first of their kind, have implications on the efficiency (or rather, inefficiency) of some commonly used SAT solving heuristics.

Our proof relies on a novel and somewhat surprising connection of the *variable space* of a proof, to the *black white* pebbling measure of an underlying graph.

[*]This research was supported by the Clore Foundation Doctoral Scholarship fund.

1 Introduction

The *resolution* proof system was initially introduced by Blake in 1937 [8], and further developed in the influential work of Robinson [32] and Davis and Putnam [22]. This system is one of the weakest in the world of propositional proof systems, and therein lies its strength. Because of its simplicity, and because all lines in a proof are clauses, this system yields easily to proof-search algorithms. Indeed, many automated theorem proving algorithms used in practice actually search for a treelike resolution proof. The simplicity and practical usefulness of resolution made it the most investigated propositional proof system.

From a theoretical point of view, the simplicity of resolution is somewhat misleading. The most important complexity measure of a resolution proof is its minimal *size*, measured in the number of clauses appearing in a proof. This measure has received much attention, and over the past thirty years, several size lower bounds have been obtained for various families of CNF formulas, starting with the lower bound of Tseitin for *regular resolution* presented in 1968 [34], and followed by the lower bound of Haken in 1985 [24], that eventually let the way to many other such bounds (e.g. [35, 20, 6, 14, 30, 31] and many others).

A second complexity measure, closely related to the size is the minimal *width*, measured as the maximal size of a clause in the proof. This measure, introduced by [14], is actually a *space* measure, as it counts the maximal space a single clause will occupy in a proof. Surprisingly, the *width* has strong correlation to the *size*, and many size lower bounds can be derived by proving (often simpler) width lower bounds [14].

With the realization of the importance of width as a complexity measure, additional motivation was given to understanding other *space* measures of resolution proofs, and their connection to both *size* and *width*. The investigation of resolution space was initiated by [33], and several upper and lower bounds were presented for this measure [23, 1, 12]. Intuitively, the *clause space* of a proof is the maximal number of clauses one needs to keep in memory while verifying this proof, and the *variable space* is the maximal "total" space needed, where one takes into account the *width* of clauses in the memory (formal definitions appear in section 2.2).

In this paper we initiate the research of optimizing *two* of the measures at once. In particular, for treelike resolution we prove a strong negative result of the following form. There exist arbitrarily large formulas $\mathcal{T}_n$ of size n, such that:

1. $\mathcal{T}_n$ has a *linear size* treelike proof that requires only *constant clause space* to verify.

2. $\mathcal{T}_n$ has a *constant width* proof.

3. There is no treelike proof of $\mathcal{T}_n$ that has *both* small width *and* small size. Actually, for any treelike proof π of $\mathcal{T}_n$ we get the following tradeoff:

$$Width(\pi) \cdot \log Size(\pi) = \Omega(\frac{n}{\log n}).$$

4. There is no (general) proof of $\mathcal{T}_n$ that has *both* small width *and* small clause space. More explicitly, for any (general) proof π of $\mathcal{T}_n$:

$$Width(\pi) \cdot CSpace(\pi) = \Omega(\frac{n}{\log n}).$$

Let us briefly explain the meaning of the previous result. When seeking a minimal size treelike proof of $\mathcal{T}_n$ (that actually does exist), one will necessarily encounter very wide clauses. When seeking a minimal width proof (that also exists (!)), the treelike size will be of exponential size. Thus, if one tries to use a DLL heuristic that avoids wide clauses - one will have to wait very long for such a proof to emerge :-). Currently, some of the most commonly used heuristics for finding unsatisfiability proofs *do* use the minimal width heuristic. Such methods might operate quite badly on certain instances, and for no good reason, because short proofs for those instances do exist.

Building on the previous result, combined with some extra ingredients, we get an interesting tradeoff for *general* resolution proofs, of the following form. There exist arbitrarily large formulas $\mathcal{T}_n$ of size n, such that

1. $\mathcal{T}_n$ has a *linear size* proof of *constant width*.
2. There is no proof of $\mathcal{T}_n$ that has *both* small clause space *and* small size. Actually, for any proof π of $\mathcal{T}_n$ we get the following tradeoff:

$$CSpace(\pi) \cdot \log Size(\pi) = \Omega(\frac{n}{\log n}).$$

Once again, this result has implications on automated theorem proving heuristics. Any algorithm that seeks for a minimal space proof, will necessarily require exponentially large size, and for no good reason, because, being less stringy in terms of space yields an optimal size proof.

1.1 Techniques - Pebbling Contradictions

Our results come from examining a natural family of contradictions based on directed acyclic graphs, and called *pebbling contradictions*. A special case of these formulas was introduced by [28] and used by [5] for separating treelike and general resolution. Their construction was generalized by [14], who gave these formulas their name, after exposing the direct connection of the treelike size to the *pebbling* measure of the underlying graph.

Since their introduction, these formulas have been quite useful in separating various forms of resolution such as *regular* from *general* [2], and *linear* and *negative* from *general* [9].

The pebbling measure of a DAG G naturally captures the *deterministic space* required to perform a computation described by G. The notion of *nondeterministic space* is analogously described by the *black-white pebbling* measure. Since pebbling captures the notion of *space*, it seems reasonable to expect that a contradiction directly based on a graph with large pebbling measure, will result in a formula that requires large space to prove. Putting this intuition into formal statement is somewhat harder, and what we show in this paper is a connection of the proof *variable space* of a pebbling contradiction, to the *black-white pebbling measure* of G. This still leaves open several interesting question regarding the connection of the *clause space* of a pebbling contradiction to either the deterministic or non-deterministic space of the underlying graph.

1.2 Paper Organization

In the following section we give the basic definition of the relevant resolution complexity measures, as well as a definition of the pebbling measures. In section 3 we prove the size-space tradeoff for treelike resolution. Building on this, we prove similar tradeoffs for general resolution in section 4. Finally, in section 5 we discuss the application of these results to automated theorem proving procedures.

2 Definitions

2.1 Resolution

For x a Boolean variable, a *literal* over x is either x (denoted also as x^1) or $\bar{x}$ (denoted also as x^0). A *clause* is a disjunction of literals. We say that a variable x *appears* in a clause C (denoted $x \in C$) if a literal over

x is an element of C. A CNF formula is a set of clauses. Let $\mathcal{T} = \{C_1, C_2 \dots C_m\}$ be a CNF formula over n variables, a *resolution derivation* π of a clause A from $\mathcal{T}$ is a sequence of clauses $\pi = \{D_1, D_2 \dots D_S\}$ such that the last clause is A and each line D_i is either some initial clause $C_j \in \mathcal{T}$ or is derived from the previous lines using the following derivation rule, called the *resolution rule*:

$$\frac{E \vee x \quad F \vee \overline{x}}{E \vee F}$$

where $x \in \{x_1, x_2, \dots, x_n\}$ and E, F are arbitrary clauses. A resolution *proof* is a resolution derivation of the empty clause $\emptyset$.

2.2 Size, Width and Space

The graph of a derivation π, denoted $\mathbf{G}_\pi$, is a directed acyclic graph with the clauses of the derivation labeling the vertices, and for derivation steps edges are added from the assumption clauses to the consequence clause. A derivation π is called *tree-like* if G_π is a tree; we may make copies of the original clauses in $\mathcal{T}$ in order to make a proof tree-like. The *size* of the derivation π is the number of lines (clauses) in it, denoted $\mathbf{Size}(\pi)$. The *width* of a clause C is the number of literals in it, denoted $|C|$. The width of π is the maximal width of a clause in π, denoted $\mathbf{Width}(\pi)$. For $\mathcal{T}$ a CNF, $\mathbf{Min-Size}(\mathcal{T})$ is the minimal size of a proof of $\mathcal{T}$, if one exists, and ∞ otherwise. The minimal treelike size ($\mathbf{Min-TSize}(\mathcal{T})$), and minimal width ($\mathbf{Min-Width}(\mathcal{T})$) are analogously defined.

For the definition of the *space* of a derivation, it is convenient to use the following definition of a derivation, introduced by [1].

Definition 2.1 (Resolution Derivation - Alternative Definition) A configuration *is a set of clauses. A* space-proof *of a CNF $\mathcal{C}$, is a sequence of configurations $\mathcal{M}_0, \dots, \mathcal{M}_s$ such that $\mathcal{M}_0 = \emptyset$, $\mathcal{M}_s = \{\emptyset\}$ (the empty clause) and for all $t \in [s]$, $\mathcal{M}_t$ is obtained from $\mathcal{M}_{t-1}$ by one of the following rules:*

AXIOM DOWNLOAD $\mathcal{M}_t := \mathcal{M}_{t-1} \cup C$ *for some clause $C \in \mathcal{C}$;*

INFERENCE $\mathcal{M}_t \subseteq \mathcal{M}_{t-1} \cup C$ *for C obtained by a single application of the resolution rule to two clauses $D, E \in \mathcal{M}_{t-1}$.*

The *clause space* of a configuration is the number of clauses in it, and its *variable space* is the sum of sizes (widths) of the clauses. The clause (variable) space of a set of configurations is the maximal clause (variable) space of the set.

Given a proof π and its graph G_π, a space proof conforming with π is a space proof with the added restriction that each inference corresponds to some node of the graph, i.e. if C is the clause inferred at time t, then $C \in \pi$ and both its assumptions appear in $\mathcal{M}_{t-1}$ [1]. The *clause space* of π, denoted $\mathbf{CSpace}(\pi)$ is the minimal clause space of a space-proof conforming with π. The *variable space* of π, $\mathbf{VSpace}(\pi)$ is analogously defined. Finally, the clause space of a CNF $\mathcal{T}$, denoted $\mathbf{Min-CSpace}(\mathcal{T})$ is the minimal clause space of a proof π of $\mathcal{T}$. The variable space of $\mathcal{T}$, $\mathbf{Min-VSpace}(\mathcal{T})$, is analogously defined.

Using the above notation, D, E are called the *assumptions* and C the *conclusion* of the inference step. If a clause F appears in $\mathcal{M}_{t-1}$ and $\mathcal{M}_t$, we say its appearance in $\mathcal{M}_{t-1}$ is an *immediate predecessor* of its appearance in $\mathcal{M}_t$, and its appearance in $\mathcal{M}_t$ is a *immediate successor* of its appearance in $\mathcal{M}_{t-1}$. Notice that a clause in $\mathcal{M}_t$ can be both an assumption, and an immediate successor of a clause from $\mathcal{M}_{t-1}$. The

[1]For ease of notation we assume all clauses in G_π appear only once. This might not be true, and in such a case one needs to keep track of the exact appearance of a clause C in π. We remark that our definitions can be generalized to deal with this case without weakening any of our results, but we omit it for the sake of clarity.

only clause that is not an immediate succesor at time t is the conclusion of the inference. The t'th step of the proof is the transition from $\mathcal{M}_{t-1}$ to $\mathcal{M}_t$, and is denoted by $\mathcal{M}_{t-1} \rightsquigarrow \mathcal{M}_t$

We end this section with the following useful theorem of Esteban and Torán:

Theorem 2.1 *[23] For π a treelike proof of size S, $\mathbf{CSpace}(\pi) \leq \log S$.*

2.3 Restrictions

For C a clause, x a variable and $a \in \{0,1\}$, the *restriction* of C setting x to a is:

$$C|_{x=a} \overset{\text{def}}{=} \begin{cases} C & \text{if } x \text{ does not appear in } C \\ 1 & \text{if the } \textit{literal } x^a \text{ appears in } C \\ C \setminus \{x^{1-a}\} & \text{otherwise} \end{cases}$$

Similarly, $\mathcal{T}|_{x=a} \overset{\text{def}}{=} \{C|_{x=a} : C \in \mathcal{T}\}$. For $\pi = \{C_1, \dots C_s\}$ a derivation of C_s from $\mathcal{T}$ and $a \in \{0,1\}$, let $\pi|_{x=a} = \{C'_1, \dots C'_s\}$ be the restriction of π on $x = a$, defined inductively by:

$$C'_i = \begin{cases} C_i|_{x=a} & C_i \in \mathcal{T} \\ C'_{j_1} \vee C'_{j_2} & C_i \text{ was derived from } C_{j_1} \vee y \text{ and } C_{j_2} \vee \bar{y} \text{ via one} \\ & \text{resolution step, for } j_1 < j_2 < i \\ C'_j \vee A|_{x=a} & C_i = C_j \vee A \text{ via the weakening rule, for } j < i \end{cases}$$

The consequence of resolving a clause B with 1 is defined to be B. We shall assume w.l.o.g. that $\pi|_{x=a}$ does not contain the clause 1, by removing all such clauses from $\pi|_{x=a}$.

2.4 Decision Trees

Let $\mathcal{T}$ be a CNF formula over n variables and m clauses. A *search problem* for $\mathcal{T}$ is the following: given an assignment $\alpha \in \{0,1\}^n$, find a clause $C_i, 1 \leq i \leq m$ such that $C_i(\alpha) = 0$, if there is such a clause, and otherwise (i.e. $\mathcal{T}(\alpha) = 1$) answer 1.

Definition 2.2 (Decision Trees for CNF Search Problems) *A Decision Tree is a binary tree, with internal vertices labeled by variables, edges labeled by 0 or 1, and leafs labeled with the possible outputs. Every assignment to the variables defines a path through the tree in the natural way, and the label at the end of the path is said to be the output of the decision tree on that assignment.*

We say that D is a Decision Tree for the Search problem for $\mathcal{T}$ if it correctly solves it on every input.

for $\mathcal{T}$ a CNF formula, Let $S_D(\mathcal{T})$ denote the minimal size of a Decision Tree solving the CNF Search Problem for $\mathcal{T}$.

Decision trees for CNF search problems are closely related to tree-like resolution, as the following lemma shows. A proof of it can be found in [13].

Lemma 2.1 *For $\mathcal{T}$ an unsatisfiable CNF, $S_T(\mathcal{T}) = S_D(\mathcal{T})$*

2.5 Pebbling

The pebbling measure of a circuit is, intuitively, the space needed for simulating the computation of the circuit on a Turing machine. For a thorough introduction to results regarding pebbling, see the excellent survey [29]. In this section we briefly state the essential definitions and facts that will be used later on in our discussion. For G a directed acyclic graph (DAG), a *source* of G is a vertex with fan-in zero, and a *sink* is a vertex with fan-out zero. Throughout this paper, a *circuit* is a DAG, with maximal fan-in 2, and a single sink.

Definition 2.3 (Pebbling) *Let G a circuit, and denote its sink by s. The pebble game on G is the following one-player game. At any point in the game, some vertices of G will have pebbles on them (one pebble per vertex), while the remaining vertices will not. A* configuration *is a subset of vertices, comprising just those vertices that have pebbles on them. The rules of the pebble game are as follows.*

1. *If all immediate predecessors of a vertex have a pebble on them, a pebble may be placed on that vertex. In particular, a pebble may be placed on a source vertex at any time.*
2. *A pebble may be removed from any vertex.*

When a pebble is placed on the sink, the player wins and the game ends. A legal *pebbling of G is a sequence of configurations $\{\mathcal{C}_0 \dots \mathcal{C}_\ell\}$, where $\mathcal{C}_0 = \emptyset$, $\mathcal{C}_\ell = \{s\}$, and $\mathcal{C}_t$ follows from $\mathcal{C}_{t-1}$ by one of the rules. The number of pebbles used in such a pebbling is the maximal size (number of pebbles) of a configuration. The pebbling price of G, denoted $\mathbf{Peb}(G)$, is the minimal number of pebbles needed in any legal pebbling of G.*

2.6 Black White Pebbling

The *black-white pebbling* measure of a graph, introduced by [18], corresponds, intuitively, to the space needed for simulating a computation of a circuit on a *non-deterministic* Turing machine. In this case, one may guess the output of a certain gate of the circuit, and then verify this guess either by computing the inputs of the gate directly, or by guessing these inputs, and then verifying their correctness. In this model, non-deterministic guesses are modeled by *white* pebbles, whereas deterministic steps are modeled by *black* ones. Once again, information about black-white pebbling can be found in [29]. In this section we briefly state the essential definitions and facts that will be used later on in our discussion.

Definition 2.4 (Black-White Pebbling) *Let G a circuit, and denote its sink by s. The black white pebble game on G is the following one-player game. At any point in the game, some vertices of G will have pebbles on them (one pebble per vertex), where some of the pebbles are* black, *and some are* white. *A* configuration *is a subset of vertices, comprising just those vertices that have pebbles on them, each labeled by B, W according to the color of the pebble. The rules of the pebble game are as follows.*

1. *A* white *pebble may be placed on any vertex.*
2. *A* black *pebble may be removed from any vertex.*
3. *If all immediate predecessors of a white-covered vertex have vertices on them, the* white *vertex can be removed.*
4. *If all immediate predecessors of a vertex have a pebble on them, a* black *pebble may be placed on that vertex.*

A legal *black-white pebbling of G is a sequence of configurations $\{\mathcal{C}_0 \dots \mathcal{C}_\ell\}$, where $\mathcal{C}_0 = \mathcal{C}_\ell = \emptyset$, there exists a time $1 \leq k < \ell$ such that $s \in \mathcal{C}_k$ (covered by either a black or a white pebble), and $\mathcal{C}_t$ follows from $\mathcal{C}_{t-1}$ by one of the rules. The number of pebbles used in such a pebbling is the maximal size (number of pebbles) of a configuration. The black-white pebbling price of G, denoted $\mathbf{BW} - \mathbf{Peb}(G)$, is the minimal number of pebbles needed in any legal black-white pebbling of G.*

We end this section by pointing out that there exist graphs with very large pebbling measures.

Theorem 2.2 *[15] For arbitrarily large n, there exist explicit constructions of circuits G_n of size n such that* $\mathbf{Peb}(G_n) = \Omega(\frac{n}{\log n})$.

Theorem 2.3 *[27] For arbitrarily large n, there exist circuits G_n of size n such that* $\mathbf{BW} - \mathbf{Peb}(G_n) = \Omega(\frac{n}{\log n})$.

3 Size-Width Tradeoffs for Treelike Resolution

In this section we prove the basic tradeoff result for treelike resolution (theorem 3.1). Our tradeoff is exposed by a family of pebbling contradictions. We start by stating the main theorem, followed by its proof. The main technical theorem of this section, theorem 3.2, gives a lower bound on the variable space of a proof of $\mathcal{T}_n$ as a function of the black-white pebbling measure of the underlying DAG.

Theorem 3.1 (Main Theorem) *For infinitely many integers n, there exist contradictions $\mathcal{T}_n$ of size n such that*

1. *$\mathcal{T}_n$ is a Horn CNF (i.e. every clause has at most one positive literal).*
2. $\mathbf{Min} - \mathbf{TSize}(\mathcal{T}_n) = O(n)$, *and* $\mathbf{Min} - \mathbf{CSpace}(\mathcal{T}_n) = O(1)$. *Moreover, there exist treelike proofs of $\mathcal{T}_n$ with linear size* and *constant space.*
3. $\mathbf{Min} - \mathbf{Width}(\mathcal{T}_n) = O(1)$.
4. *For any treelike proof π of $\mathcal{T}_n$:* $\mathbf{Width}(\pi) \cdot \log \mathbf{Size}(\pi) = \Omega(\frac{n}{\log n})$.

In the rest of this section we prove our main theorem. Our tradeoff will be exposed by the following *pebbling contradictions*, generalizing the ones introduced by [14].

Definition 3.1 (Pebbling Contradictions) *Let G be a circuit, and $k > 0$ be an integer. Associate k distinct Boolean variables $x(v)_1, \ldots, x(v)_k$ with every vertex $v \in V(G)$. Peb_G^k, the k'th degree* Pebbling Contradiction *of G is the conjunction of:*

Source axioms: $\bigvee_{i=1}^{k} x(v)_i$ *for each source v.*

Sink axioms: $\bar{x}(s)_i$ *for s the sink of G, and $i \in [k]$.*

Pebbling axioms: $\bar{x}(u)_i \vee \bar{x}(w)_j \vee x(v)_1 \vee \ldots \vee x(v)_k)$ *for u, w the two predecessors of v, and $i, j \in [k]$.*

In this paper we will only be interested in $k = 1$ or $k = 2$, but our results generalize to larger k. Theorem 3.1 will be derived by using Peb_G^1 where G is a graph with large black white pebbling measure. Such graphs exist, by theorem 2.3. As we prove various properties of these pebbling contradictions, we will refer the reader to the proper part of theorem 3.1 that is being proved.

Notice that Peb_G^k is a non-satisfiable $k+2$-CNF over $k \cdot |V|$ variables, with $O(k^2 \cdot |V|)$ clauses. A CNF is said to a *Horn CNF* if every clause has at most one positive literal. It is easy to verify that Peb_G^1 is a Horn CNF. We now show that Peb_G^k has a short, width $O(k)$ resolution proof. This in particular proves part 3 of theorem 3.1.

Lemma 3.1 *For G a circuit, there exists a proof π of Peb_G^k such that* $\mathbf{Size}(\pi) = O(k^2 \cdot |V|)$ *and* $\mathbf{Width}(\pi) = \max\{2k, k+2\}$.

Proof: Fix a topological order on V. Along this order, we derive inductively for each $v \in V$ the clause

$$\bigvee_{i=1}^{k} x(v)_i \tag{1}$$

If v is a source in G, then (1) is a source axiom. If v has two predecessors, u and w, we have by induction derived $\bigvee_{i=1}^{k} x(u)_i$ and $\bigvee_{i=1}^{k} x(w)_i$. Together with the k^2 Pebbling axioms for v, these imply (1). By the implicational completeness of resolution there is a derivation of (1) from the above mentioned clauses. It is not hard to verify that this derivation can be done in $k(k+1)$ resolution steps, with width $\max\{2k, k+2\}$. Hence, starting from the source axioms, one can infer (1) for a target $v \in T$, and then, resolving with the proper target axioms, derive 0. □

The following lemma proves part 2 of theorem 3.1.

Lemma 3.2 *There exists a treelike proof of Peb_G^1 with $\mathbf{Size}(\pi) = O(|V|)$ and $\mathbf{CSpace}(\pi) = O(1)$.*

Proof: We use the equivalence of decision trees and resolution (lemma 2.1) and show that the CNF search problem over Peb_G^1 has linear size. Notice that in the 1st degree pebbling contradiction, every vertex has a unique variable associated with it, so for simplicity we identify the two. We use the term *vertex* to denote a vertex of G, and the term *node* to denote a vertex of the decision tree we are forming. Thus, at every *node* of the decision tree we query a *vertex* of G.

Define a topological order on the vertices of G, and *query* vertices according to this order. If one of the answers leads to a falsifying assignment to some clause, label this answer by the falsified clause, and proceed via the other answer. If both answers lead to a falsifying assignment, stop.

Since we are querying variables in a topological order, when we query v we have already queried its predecessors earlier.

Claim 3.1 *When a vertex v is queried, answering 0 leads to a falsifying assignment.*

Proof: By induction on the topological order. For v a source, the claim is trivial, because answering 0 falsifies the source axiom x_v. Assume v is not a source. By the topological ordering, we have already queried its predecessors (say u, w). By induction, we must have answered both queries by 1, because answering 0 would lead to a falsifying assignment. Hence we already have $u = w = 1$. Answering 0 to the query on v leads to a falsifying assignment to the pebbling axiom $\bar{x}_u \vee \bar{x}_w \vee x_v$. The claim is proved. □

Finally, when we query the sink s, answering 1 falsifies the sink axiom, and hence both answers falsify a clause, and the decision tree is complete.

Since at least one answer of every node is a leaf, and we query every vertex at most once, we conclude that the resulting decision tree has linear size, and has constant space. The lemma is proved. □

3.1 Essential Clauses and Proofs

We have almost completed the proof of theorem 3.1, leaving out part 4. We now deal with this part. In order to prove our lower bound, in theorem 3.2, we will need some technical introductory insights about the structure of proofs.

Definition 3.2 (Essential proofs) *For π a proof of a CNF $\mathcal{C}$, and $\{\mathcal{M}_1 \ldots \mathcal{M}_\ell\}$ a space-proof conforming to π, we define* essential clauses *in memory configurations by backward induction.*

1. *Let ℓ be the first time the empty clause $\emptyset$ appears in a memory configuration. Then $\emptyset$ is* essential *at time ℓ.*

2. *If C is an essential clause at time t, and was inferred at time t from the assumptions D, E, then D, E are essential at time $t-1$.*

3. *If C is essential at time t then its immediate predecessor, if it exists (i.e. if C is not an axiom downloaded at time t), is essential at time $t-1$.*

A space-proof is called essential *if all clauses in it are essential at all times.*

Lemma 3.3 *Removing non-essential clauses from a space-proof conforming with π gives a space-proof that conforms with π, without increasing the clause space , variable space, width or size.*

Using lemma 3.3 we can assume from now on that all our proofs are essential. Here are a few useful properties of essential proofs. For $\mathcal{M}$ a set of clauses, define $Vars(\mathcal{M})$ to be the set of variables appearing in clauses of $\mathcal{M}$.

Lemma 3.4 *For $\pi = \{\mathcal{M}_0, \dots, \mathcal{M}_\ell\}$ an essential space-proof of $\mathcal{C}$:*

1. *ℓ is the first time $\emptyset$ appears in a memory configuration, and $\mathcal{M}_\ell = \{\emptyset\}$.*

2. *If x is the variable resolved upon in the inference step $\mathcal{M}_{t-1} \rightsquigarrow \mathcal{M}_t$, then*
$$Vars(\mathcal{M}_{t-1}) - \{x\} \subseteq Vars(\mathcal{M}_t) \subseteq Vars(\mathcal{M}_{t-1}).$$

3.2 Proof of Part 4 of Theorem 3.1

We are ready to prove the main technical theorem of this section.

Theorem 3.2 $\mathbf{Min} - \mathbf{VSpace}(Peb^1_G) \geq BW - Peb(G)$.

Before presenting the proof, let us demonstrate how this theorem implies part 4 of theorem 3.1. By theorem 2.3, there exist graphs of size n with black white pebbling measure $n/\log n$. Let G be such a graph. By theorem 2.1, if Peb^1_G has a treelike proof of size S then this proof has clause space $\leq \log S$. By theorem 3.2 such a proof must have width $\geq \frac{n}{\log S \cdot \log n}$. Part 4 is proved. We now proceed to a proof of theorem 3.2.

Proof: We claim that any essential proof of Peb^1_G, after minor modifications, is actually a black white pebbling of G. Let $\pi = \{\mathcal{M}_0, \dots, \mathcal{M}_\ell\}$ be an essential space-proof of Peb^1_G. For $\mathcal{M}$ a set of clauses, define $Positive(\mathcal{M}) \subseteq Vars(\mathcal{M})$ to be the set of variables that have at least one appearance as a positive literal in $\mathcal{M}$. Define the following pebbling sequence on G.

$$\mathcal{C}_t(v) = \begin{cases} Black & x_v \in Positive(\mathcal{M}_t) \\ White & x_v \in Vars(\mathcal{M}_t) - Positive(\mathcal{M}_t) \\ Null & \text{otherwise} \end{cases}$$

By its definition, the sequence $\{\mathcal{C}_0 \dots \mathcal{C}_\ell\}$ uses no more pebbles than $\mathbf{VSpace}(\pi)$, so we only need to show that $\{\mathcal{C}_0 \dots \mathcal{C}_\ell\}$ is a legal BW pebbling sequence of G. This we do by induction on $t = 0 \dots \ell$, and split the proof into cases.

Inference steps: Assume the variable x_v was resolved upon in the t'th step. By lemma 3.4, the only change in the set of literals in the t'th step, would be focused on x_v. We know that $v \in Black(\mathcal{C}_{t-1})$ because $x_v \in Positive(\mathcal{M}_{t-1})$. Let us examine the possible cases. If $x_v \notin Vars(\mathcal{M}_t)$ we can remove the black pebble from v. If $x_v \in Positive(\mathcal{M}_t)$, then we leave the black pebble on v at time t. Finally, if $x_v \in Vars(\mathcal{M}_t) - Positive(\mathcal{M}_t)$, we remove the black pebble from v, and place a white pebble on it. All steps are legal BW pebbling steps, showing that the transition from $\mathcal{C}_{t-1}$ to $\mathcal{C}_t$ is legal.

Axiom Downloads: We may need to place new pebbles on G, and check that this is done in a legal way. Let A be the axiom downloaded. If A is a source axiom, we need to place a black pebble on a source vertex v. This can always be done, for even if v is covered by a white pebble, we may remove this pebble (v is a source) and place a black pebble instead. Similarly, if v is the sink, we wish to place a white pebble on it, only if v is not covered by a black pebble at time $t-1$, and this is clearly a legal step. Finally, if $A = \bar{x}_u \vee \bar{x}_w \vee x_v$ we wish to place a black pebble on v and white pebbles on u, w. If u is covered by some pebble (black or white), we don't need to do a thing, and the same applies to w. If one of u, w is uncovered, we place a white pebble on it, which is legal. Now notice that u, w are covered, so one can remove any white pebble from v, if such a pebble exists, and place a black pebble on it. We have showed how to derive our new configuration via legal pebbling steps.

So far we have shown that all steps in $\{\mathcal{C}_0 \ldots \mathcal{C}_\ell\}$ are legal. By the soundness of resolution, any proof of Peb^1_G must use the sink axiom $\bar{x}_s$, because $Peb^1_G - \{\bar{x}_t\}$ is satisfiable. Hence, at some point in our pebbling sequence a pebble will be placed on s. Since $Vars(\mathcal{M}_0) = Vars(\mathcal{M}_\ell) = \emptyset$ we know that $\mathcal{C}_0 = \mathcal{C}_\ell = \emptyset$ and we conclude that $\{\mathcal{C}_0 \ldots \mathcal{C}_\ell\}$ is a legal BW pebbling of G. The theorem is proved. $\square$

4 Size-Space Tradeoffs for General Resolution

In this section, we present a general transformation that takes *any* CNF that requires large variable space and forms a new formula that has a large size-space tradeoff. The trick of replacing each variable by two variables was exploited in [14] in order to obtain an exponential separation of treelike and general resolution. The form we use here, was first introduced by Alekhnovich and Razborov [4], to transform a CNF $\mathcal{T}$ with large minimal *width* into a CNF $\mathcal{T}'$ with large minimal proof size, exponential in the width of $\mathcal{T}$ (see theorem 4.1) Applying this transformation to our pebbling formula from the previous section, we will get our results. The main theorem of this section is theorem 4.2.

Definition 4.1 (Xorification) *For $\mathcal{T}(x_1, \ldots, x_n)$ a CNF, we define its* xorification *to be the following formula over variables $\{x'_1, x''_1, x'_2, x''_2, \ldots, x'_n, x''_n\}$. The* xorification *of a literal $x^a_i, a \in \{0,1\}$, is the formula $\mathcal{X}(x_i) \overset{\text{def}}{=} x'_i \oplus x''_i \oplus a$. The* xorification *of a clause $\bigvee_{i \in I} \ell_i$ is the CNF formula equivalent to $\bigvee_{i \in I} \mathcal{X}(\ell_i)$, and the* xorification *of $\mathcal{T}$, denoted $\mathcal{X}(\mathcal{T})$ is the conjunction of the xorifications of the clauses.*

If our original CNF is not too wide, then the xorification does not blow up its size by too much.

Lemma 4.1 *If $\mathcal{T}$ is a k-CNF with m clauses and n variables, then $\mathcal{X}(\mathcal{T})$ is a $2k$-CNF with $\leq 2^k \cdot m$ clauses and $2n$ variables.*

As mentioned above, the xorification was first introduced by Alekhnovich and Razborov, who proved the following theorem.

Theorem 4.1 *[4] For any k-CNF $\mathcal{T}$, $\mathbf{Min-Size}(\mathcal{X}(\mathcal{T})) = \exp(\Omega(\mathbf{Min-Width}(\mathcal{T}) - k))$.*

We notice that using the xorification, and essentially the same proof as that of the previous theorem, one gets the following size space tradeoff for general resolution.

Theorem 4.2 *For any CNF $\mathcal{T}$ and any resolution proof π of $\mathcal{X}(\mathcal{T})$: $\mathbf{CSpace}(\pi) \cdot \log_{\frac{4}{3}} \mathbf{Size}(\pi) \geq \mathbf{Min-VSpace}(\mathcal{T})$*

Proof: Given a proof π of $\mathcal{X}(\mathcal{T})$, we apply to it the following random restriction ρ. For each pair x'_i, x''_i independently select one of the pair, with probability $\frac{1}{2}$, and set it to $\{0,1\}$ with probability $\frac{1}{2}$. It is easy to see that $\mathcal{X}(\mathcal{T})|\rho$ is equivalent to $\mathcal{T}$ up to renaming of variables and literals. Look at a clause $C \in \pi$. We claim that $\mathbf{P}[C|_\rho \neq 1] \leq \frac{3}{4}^{|C|}$. To see this notice that if only one of the pair x'_i, x''_i appears in C, the probability that this literal is set to 1 is $\frac{1}{4}$. If both variables appear (as some literal) then the probability that this pair is not set to 1 is $\frac{1}{2} < \frac{3}{4}^2$. For any w, the expected number of clauses remaining in π after applying ρ, and having width $\geq w$, is by linearity of expectation, $\mathbf{Size}(\pi) \cdot \frac{3}{4}^{-w}$. Set $w = \frac{\mathbf{Min-VSpace}(\mathcal{T})}{\mathbf{CSpace}(\pi)}$, and conclude that if $\mathbf{Size}(\pi) < \frac{3}{4}^w$, then there exists a restriction ρ for which $\pi|_\rho$ has width w. Notice that clause space does not increase under restrictions, and hence $\mathbf{Width}(\pi|_\rho) \cdot \mathbf{CSpace}(\pi|_\rho) < \mathbf{Min-VSpace}(\mathcal{T})$, in contradiction with theorem 3.2. We conclude that $\log_{\frac{4}{3}} \mathbf{Size}(\pi) \geq \frac{\mathbf{Min-VSpace}(\mathcal{T})}{\mathbf{CSpace}(\pi)}$ and the theorem is proved. $\square$

We can now state our size - space tradeoffs for general resolution:

Theorem 4.3 *For infinitely many integers n, there exist contradictions $\mathcal{T}_n$ of size n such that*

1. $\mathbf{Min-Width}(\mathcal{T}_n) = O(1)$, *and* $\mathbf{Min-Size}(\mathcal{T}_n) = O(n)$. *Moreover, there exists a proof of $\mathcal{T}_n$ of linear size* and *constant width.*
2. $\mathbf{Min-VSpace}(\mathcal{T}_n) = \Omega(\frac{n}{\log n})$.
3. *For any general proof π of $\mathcal{T}_n$:* $\mathbf{CSpace}(\pi) \cdot \log \mathbf{Size}(\pi) = \Omega(\frac{n}{\log n})$.

Proof: Take $\mathcal{T}_n = \mathcal{X}(Peb^1_G)$ for G a graph over n vertices with black white pebbling measure of $\frac{n}{\log n}$. Part 1 is immediate from the proof of lemma 3.1, that can be applied directly to our $\mathcal{T}_n$. Part 2 follows from the fact that any restriction that fixes exactly one of each pair x'_i, x''_i reduces $\mathcal{T}_n$ to Peb^1_G. Since a restriction does not increase the variabe space, this proves part 2. Finally, part 3 follows immediately from theorem 4.2. $\square$

5 Discussion and Applications

5.1 Finding Minimal Width Proofs

The size-width relations of [14] show that if $\mathcal{T}$ has a treelike proof of size S, then it has a (perhaps different) proof of width $\log S$. Similar, but slightly weaker results hold also for general resolution. The proof of this theorem uses a transformation of a small proof to a small width proof. In this process, as it was presented in [14], the size of the proof might grow exponentially. It is only natural to wonder whether this blowup is essential, or perhaps a better transformation exists, that retains small size while producing small width.

Our results give a negative answer to this question. Indeed, the transformation of a small proof of Peb^1_G into a proof with small width, must increase the treelike size exponentially, for otherwise it would result in a proof with small variable space (recall that the clause space is upper bounded by $\log \mathbf{Size}(\pi)$, theorem 2.1). Even if one is willing to settle for a small size *general* proof, one will pay a price in terms of clause space, as our results show.

5.2 The Restricted Width Algorithm

A reasonable heuristic for finding short resolution proofs is to restrict the width, and look for small width proofs. The upper bounds of [14] give additional theoretical motivation to such a heuristic. Our results show that for certain cases, this method will operate poorly. If one seeks a minimal width proof of Peb^1_G, then necessarily the clause space will be large, and hence also the treelike size. If, once again, one is willing to settle for general proofs, then at least we can say that he will pay a large price in terms of the clause space needed.

5.3 Other SAT Solving Heuristics

Some of the most extensively used and investigated methods for proving unsatisfiability of CNF formulas, are called Davis-Putnam procedures. Actually, these procedures are derived from a system devised by Davis, Logemann and Loveland [21], and hence we will refer to them as *DLL Procedures*. A DLL procedure relies on choosing a variable x, and trying to refute $\mathcal{T}|_{x=0}$ and $\mathcal{T}|_{x=1}$ recursively.

If $\mathcal{T}$ is unsatisfiable, $DLL(\mathcal{T})$ terminates providing a tree-like resolution proof of $\mathcal{T}$. The DLL procedure is actually a family of algorithms, diversified by the different strategies for picking the next variable x to be queried.

All DLL heuristics use the *unit clause rule*, which selects x to be variable appearing in some clause of width 1, if such a clause exists. This heuristic is good since at least one assignment to x will end in a falsified clause (indeed, our minimal treelike proof for Peb^1_G follows this heuristic. See proof of lemma 3.2).

A generalization of this heuristic is the *minimal clause rule*, which selects a variable x that appears in a clause of minimal *width*. A different heuristic that might seem tempting, especially given the size-width tradeoff, is to look for a small treelike resolution proof that has also small width. Our results show this is not a good heuristic, as in certain cases, trying to maintain minimal width in the proof, results in an exponential blowup of the treelike resolution size.

Finally we point out that if one applies the minimal clause heuristic to Peb^1_G, one still gets an optimal size proof. The interested reader may easily verify this. Does this mean that this heuristic is necessarily good ? the following example, based on a trivial graph construction, gives a negative answer to this question.

Theorem 5.1 *For all n there exist contradictions $\mathcal{T}_n$ of size $O(n)$, with the following properties:*

1. $\mathbf{Min-TSize}(\mathcal{T}_n) = O(n)$.
2. *Any DLL procedure using the minimal clause heuristic will have running time $2^{\Omega(n)}$*

Proof: Look at the depth n *X-DAG* defined as follows. This is a layered graph, in each layer two vertices. We number the vertices v_i^1, v_i^2 for $i = 1 \dots n$. There are two sources: v_1^1, v_1^2. For $i > 1, b \in \{1,2\}$ the predecessors of v_i^b are v_{i-1}^1, v_{i-1}^2. The sink s has predecessors v_n^1, v_n^2. This graph has $2n+1$ vertices. For G an X-DAG, take Peb^1_G, and replace each source clause $x_{v_1^b}$ (for $b \in \{1,2\}$) by (say) the CNF

$$\bigwedge_{\epsilon \in \{0,1\}^4} ((\bigvee_{i=1}^{4} y_i^{\epsilon_i}) \vee x_{v_1^b})$$

(any CNF that is equivalent to $x_{v_1^b}$, and has derivation width > 3 would do). Denote this formula by $\mathcal{T}_n$. We start with an upper bound on arbitrary treelike size.

Claim 5.1 $\mathbf{Min-TSize}(\mathcal{T}_n) = O(n)$

Proof: Look at the following decision tree. Query $x_{v_1^1}$. If answered 0, query all y variables, to derive a contradiction of size $\leq 2^4$. If answered 1, query $x_{v_1^2}$. If answered 0, query once again all y variables to derive small contradiction, otherwise we have obtained $x_{v_1^1} = x_{v_1^2} = 1$, and now, under this restriction, $\mathcal{T}_n$

becomes a 1st degree pebbling contradiction, which by lemma 3.2, has a small decision tree. The claim is proved. □

Our lower bound for the minimal clause heuristic will follow immediately from the following claim. Let $Time(\mathcal{T})$ denote the running time of the minimal clause heuristic DLL on $\mathcal{T}$. Notice that $\mathcal{T}_n$ has a single unit clause $\bar{x}_s$, and hence our DLL starts by querying the variable x_s. Our lower bound follows immediately from the following claim.

Claim 5.2 $Time(\mathcal{T}_n|_{x_s=0}) \geq 2^n$.

Proof: By induction on n. For $n = 0$, $\mathcal{T}_0|_{x_s=0} = \emptyset$, and thus a proof has size 1. For the induction step, notice that $\mathcal{T}_n|_{x_s=0}$ has only one clause of width 2, namely $\bar{x}_{v_n^1} \vee \bar{x}_{v_n^2}$, and all other clause have width ≥ 3. Hence we query wlog $x_{v_n^1}$. If we answer 0, our new formula is simply $\mathcal{T}_{n-1}|_{x_s=0}$ where the sink $s = v_n^1$, and we apply induction. If we answer 1, we have now a single unit clause, namely $\bar{x}_{v_1^2}$, and we must query this variable. Answering 1 leads to a falsified clause, whereas answering 0 leads once again to $\mathcal{T}_{n-1}|_{x_s=0}$, this time with $s = v_n^2$. Once again we apply induction. The claim is proved, and with it the theorem. □

6 Open Problems

What is the minimal clause space of Peb_G^2 ?

7 Acknowledgements

We thank Nicola Galesi for helpfull comments on an earlier version of this paper.

References

[1] M. Alekhnovich, E. Ben-Sasson, A. A. Razborov, A. Wigderson. Space complexity in propositional calculus. In *Proceedings of the* 32*nd STOC*, pages 358–367, 2000.

[2] M. Alekhnovich, J. Johannsen, T. Pitassi, A. Urquhart An Exponential Separation between Regular and General Resolution Found at *Electronic Colloqium on Computational Complexity*, Reports Series 2001, Available at http://www.eccc.uni-trier.de/eccc/. Technical Report TR01-056

[3] M. Alekhnovich, A. A. Razborov. Resolution is Not Automatizable Unless W[P] is Tractable. In *Proceedings of the* 42*nd IEEE FOCS* , 2001.

[4] M. Alekhnovich, A. A. Razborov. Personal Communication, Institute for Advanced Study, 2001.

[5] M. L. Bonet, J. L. Esteban, N. Galesi, J. Johannsen. Exponential Separations between Restricted Resolution and Cutting Planes Proof Systems. submitted.

[6] P. Beame, R. Karp, T. Pitassi, M. Saks. On the Complexity of Unsatisfiability Proofs for Random k-CNF Formulas. Submitted.

[7] P.Beame, T. Pitassi. Simplified and Improved resolution lower bounds. In *37th Annual Symposium on Foundations of Computer Sciecne*, pp 274-282. Burlington, VT, October 1996. IEEE.

[8] A. Blake. Canonical Expressions in Boolean Algebra. PhD Thesis, University of Chicago, 1937.

[9] J. Buresh-Oppenheim, D. Mitchell, T. Pitassi. Linear and Negative Resolution are Weaker than Resolution. Found at *Electronic Colloqium on Computational Complexity*, Reports Series 2001, Available at http://www.eccc.uni-trier.de/eccc/. Technical Report TR01-074

[10] S. Buss, T. Pitassi. Resolution and the Weak Pigeonhole Principle. Typeset manuscript, to appear in *CSL97*.

[11] S.R. Buss, G. Turán. Resolution Proof of generalized Pigeonhole principles. In *Theoretical Comp. Sci.*, 62:(1988)311-317.

[12] E. Ben-Sasson and N. Galesi. Space Complexity of Random Formulae in Resolution. In *Proceedings of the 16th IEEE Conference on Computational Complexity*, 2001.

[13] E. Ben-Sasson, R. Impagliazzo and A. Wigderson. Near-Optimal Separation of General and Tree-Like Resolution. Technical Report TR00-005, Electronic Colloquium on Computational Complexity, 2000.

[14] E. Ben-Sasson, A. Wigderson. Short Proofs are Narrow - Resolution made Simple. In *STOC99*.

[15] R. Celoni, W.J. Paul, R.E. Tarjan Space Bounds for a Game on Graphs. In *Math. Systems Theory*, 10 (1977), pp 239-251.

[16] M. Clegg, J. Edmonds, R Impagliazzo Using the Groebner Basis algorithm to find proofs of unsatisfiability In *Proceedings of the 28th ACM symposium on Theory of Computing, 1996*, pp 174-183.

[17] S. A. Cook. An Observation on Time-Storage Trade-Off. In *J. of Comp. and Sys. Sci.*, Vol 9 (1974), pp. 308-316.

[18] S.A. Cook, R. Sethi. Storage Requirements for Deterministic Polynomial Time Recognizable Languages, In *J. Comp. and Sys. Sci. 13* (1976) pp. 25-37.

[19] S. A. Cook, R. Reckhow. The relative efficiency of propositional proof systems. In *J. of Symbolic Logic*, Vol. 44 (1979), pp. 36-50.

[20] V. Chvátal, E. Szemerédi Many Hard Examples for Resolution In *J. of the ACM*, Vol 35 No. 4, pp. 759-768.

[21] M. Davis, G. Logemann, D. Loveland. A Machine program for theorem proving. In *Communications of the ACM*, 5:394-397, 1962.

[22] M. Davis, H. Putnam. A computing procedure for quantification theory. In *Journal of the ACM, 7* pp. 201-215, 1960. Reprinted in *Automation of reasoning,* vol. 1, by J. Siekmann and G. Wrightson, ed. Springer-Verlag, Berlin, 1983.

[23] J. L. Esteban, J. Torán. Space bounds for Resolution. In *Proceedings of the* 16*th* STACS, pages 530–539, 1999.

[24] A. Haken. The Intractibility of Resolution. In *Theoretical Computer Science*, 39 (1985), pp. 297-308.

[25] J.E. Hopcroft, W. Paul, L. Valiant. On Time vs. Space. In *J. ACM*, 24 (1977), pp. 332-337.

[26] R. Impagliazzo, P Pudlák, J. Sgall. Lower Bounds for the Polynomial-Calculus and the Groebnes Basis Algorithm. Found at *Electronic Colloqium on Computational Complexity*, Reports Series 1997, Available at http://www.eccc.uni-trier.de/eccc/. Technical Report TR97-042.

[27] T. Lengauer, R. E. Tarjan. Upper and Lower Bounds on Time-Space Tradeoffs, In *11th ACM Symp. on Theory of Computing*, (1979) pp. 262-277.

[28] R. Raz, P. McKenzie. Separation of the Monotone NC Hierarchy. submitted.

[29] N. Pippenger Pebbling. *Technical Report*, IBM Watson Research Center.

[30] T. Pitassi, R. Raz. Regular Resolution lower bounds for the Weak Pigeonhole Principle. In *Proceedings of the 33rd ACM STOC*, 2001.

[31] R. Raz. Resolution Lower Bounds for the Weak Pigeonhole Principle. Found at *Electronic Colloqium on Computational Complexity*, Reports Series 2001, Available at http://www.eccc.uni-trier.de/eccc/. Technical Report TR97-021.

[32] J. A. Robinson A machine-oriented logic based on the resolution principle. In *Journal of the ACM*, vol. 12, pp. 23-41. Reprinted in *Automation of reasoning*, vol. 1, by J. Siekmann and G. Wrightson, ed. Springer-Verlag, Berlin, 1983.

[33] J. Torán. Lower Bounds for Space in Resolution. In *Proceedings of CSL 1999*, pages 362–373. 1999.

[34] G.S. Tseitin On the Complexity of Derivation in Propositional Calculus. In *Studies in Constructive Mathematics and Mathematical Logic*, Part 2. Consultants Bureau, New-York-London, 1968, pp. 115-125.

[35] A. Urquhart. Hard Examples for Resolution. In *J. of the ACM*, Vol 34 No. 1, pp. 209-219.