

Size Space tradeoffs for Resolution

Eli Ben-Sasson*
Department of Computer Science
Technion, Haifa, Israel.
eli@cs.technion.ac.il

March 1, 2009

Abstract

We investigate tradeoffs of various basic complexity measures such as *size*, *space* and *width*. We show examples of 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 have implications to the efficiency (or rather, inefficiency) of some commonly used SAT solving heuristics.

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

1 Introduction

The *resolution* proof system was introduced by Blake in 1937 [Bla37], and further developed in the influential work of Robinson [Rob65] and Davis and Putnam [DP60]. 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 resolution proof. The simplicity and practical usefulness of resolution has 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 formulas in conjunctive normal form (referred to henceforth as CNF formulas, or, simply, CNFs), starting with the lower bound of Tseitin for a (strict) subsystem of resolution known as *regular resolution* presented in [Tse68], and followed by the lower bound of Haken in [Hak85], that eventually let the way to many other such bounds (e.g. [Urq87, CS88, BKPS02, BSW01, PR04, Raz04] 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, suggested by Galil in [Gal77], is actually a *space* measure, as it counts the maximal space a single clause will occupy in a proof. The width measure has strong correlation to the *size* and many size lower bounds can be derived by proving (often simpler) width lower bounds [BSW01].

*Research supported by an Alon fellowship of the Israeli council for higher education and by grants from the Israeli Science Foundation and the US-Israel Binational Science Foundation. An initial version of this paper was presented in the 34th Annual ACM Symposium on Theory of Computing (STOC '2002) [BS02].

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 [Tor99], and several upper and lower bounds were presented for this measure [ET01, ABSRW02, BSG03]. 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 2.4).

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 small width *and* small size simultaneously. Actually, for any treelike proof π of \mathcal{T}_n we get the following tradeoff:

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

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

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

Let us briefly explain the meaning of the previous result. When seeking a treelike proof of \mathcal{T}_n that is of small size (and such a proof does exist), one will necessarily encounter very wide clauses. When seeking instead a proof of small width (that also exists), the treelike size will be exponential. Thus, if one tries to use a DLL heuristic that avoids wide clauses - one will have to wait a very long time for a proof to be found. Currently, some algorithms 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 a separation between width efficiency and simultaneous size-space efficiency for *general* resolution proofs. In particular, we show that 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 small clause space *and* small size simultaneously. Actually, for any proof π of \mathcal{T}_n we get

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

The result above has been recently subsumed by [BSN08] who showed that any proof of the above mentioned formula must require clause space $\Omega(n/\log n)$, which implies part 2 above. However, we include this result because its proof differs from, and is arguably simpler than, that of [BSN08].

1.1 Techniques - Pebbling Contradictions

Our results come from examining a natural family of contradictions based on directed acyclic graphs (DAGs) called *pebbling contradictions*. A special case of these formulas was introduced by [RM99] and used by [BEGJ00] for separating treelike and general resolution. Their construction was generalized by [BSW01], who gave these formulas their name and exposed 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 proving lower bounds on clause space [Nor06, HP07, NH08, BSN08] and in separating various forms of resolution such as *regular* from *general* [AJPU02], and *linear* and *negative* from *general* [BOMP01].

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 .

1.2 Subsequent developments

A question left completely open by the preliminary version of this paper was that of the relationship between proof size and clause space. In a nutshell, this question asks whether there exist contradictions of size n that have proofs of size $O(n)$ but whose clause-space is super-constant. Initial progress on this question was reported in [Nor06] by showing that for (pebbling contradictions over) binary trees, the clause space is indeed lower bounded by the black-white pebbling measure of the tree, which is equal to its depth. This implied that the clause space could, in certain cases, be as large as $\Omega(\log n)$. This result was improved by [NH08] who showed the clause space could be as large as $\Omega(\sqrt{n})$. Recently, [BSN08] resolved the question by showing that clause space could be as large as $\Omega(n/\log n)$. This answer is tight up to a multiplicative constant because by a result of [HPV77], every DAG of size m has a pebbling cost of at most $O(m/\log m)$ and this result directly implies that a resolution proof of size $O(n)$ can be carried out in space $O(n/\log n)$.

1.3 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 Pebbling

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. The pebbling measure of a circuit, introduced in [WS72], 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 surveys [Pip80, Nor09]. In this section we briefly state the essential definitions and facts that will be used later on in our discussion.

Definition 2.1 (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 new pebble may be placed on that vertex. In particular, a pebble may be placed on a source vertex at any time.
2. If all immediate predecessors of a vertex v have a pebble on them, a pebble may be moved from one immediate predecessor to v .
3. 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 .

Remark 2.2. The “sliding” rule above (rule # 2) is sometimes omitted in definitions of the pebbling game. It was shown by [BL78] that the pebbling price of a graph without the sliding rule is exactly greater by 1 than its cost when allowing use of this rule. We choose to include the sliding rule to simplify the definition of clause space later on in Section 2.4.

2.2 Black-White Pebbling

The *black-white pebbling* (*BW-pebbling*) measure of a graph, introduced by [CS76], 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 [Pip80]. In this section we briefly state the essential definitions and facts that will be used later on in our discussion.

Definition 2.3 (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 v have pebbles on them, the *white* pebble can be removed from v .
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 $\{C_0 \dots C_\ell\}$, where $C_0 = C_\ell = \emptyset$, there exists a time $1 \leq k < \ell$ such that $s \in C_k$ (covered by either a black or a white pebble), and C_t follows from 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 .

2.3 Resolution

For x a Boolean variable, a *literal* over x is either x or \bar{x} . The former is called a *positive* literal and often denoted by x^1 and the latter is a *negative* literal and is often denoted by 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 and its *size* is the cardinality of the set. A CNF is said to be a *Horn CNF* if every clause has at most one positive literal. 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*:

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

where $x \in \{x_1, x_2, \dots, x_n\}$ and E, F are arbitrary clauses. The clauses $E \vee x, F \vee \bar{x}$ are known as the *assumption clauses* of the inference and the clause $E \vee F$ is its *consequence clause*. A *resolution proof* is a resolution derivation of the empty clause $\mathbf{0}$. Often, resolution is defined to include the additional *weakening* rule which allows to deduce $A \vee \ell$ from A , where ℓ is any literal. Adding the weakening rule does not affect any of the proof-complexity measures we study in this paper (namely, size, space or width) so we focus on resolution without weakening and notice all results apply to resolution with weakening just as well.

2.4 Size, Width and Space

For $\pi = \{D_1, \dots, D_S\}$ a resolution derivation from a CNF \mathcal{T} , a directed acyclic graph G_π is said to be a *derivation graph* of π if the following four conditions hold.

- G_π has S vertices.
- There exists a topological ordering v_1, \dots, v_S of the vertices of G_π , and a labeling of vertices by clauses such that $\{\ell(v_1), \dots, \ell(v_S)\} = \pi$, where $\ell(v_i)$ is the clause that labels v_i .
- If v is a leaf then $\ell(v)$ is an axiom of \mathcal{T} .
- If u, w are the immediate predecessors of v then $\ell(u), \ell(w)$ are assumption clauses and $\ell(v)$ is the consequence clause of a resolution inference.

Notice that a derivation π may have several different derivation graphs associated with it. A derivation π is called *tree-like* if it has a derivation graph G_π that is a tree. 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 $\mathbf{Width}(\pi)$. The *clause space* of a derivation π is the minimal pebbling cost of a derivation graph associated with π . For \mathcal{T} a CNF, let $\mathbf{Min} - \mathbf{Size}(\mathcal{T})$ be the minimal size of a proof of \mathcal{T} , if one exists, and ∞ otherwise. The minimal tree-like size $\mathbf{Min} - \mathbf{TSize}(\mathcal{T})$, minimal width $\mathbf{Min} - \mathbf{Width}(\mathcal{T})$ and minimal clause space $\mathbf{Min} - \mathbf{CSpace}(\mathcal{T})$ are analogously defined.

At times it will be convenient to use the alternative definition of space introduced by [ABSRW02]. This definition will also allow us to define *variable space* below.

Definition 2.4 (Resolution Derivation — space-oriented 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 $\mathcal{M}_0 = \emptyset$, $\mathcal{M}_s = \{\mathbf{0}\}$ (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 clauses D, E are called the *assumptions* and C the *conclusion* of the inference step.

Remark 2.5. The definition in [ABSRW02] includes an additional rule, called erasure, which allows to remove clauses from memory. The definition of [ABSRW02] also requires that an inference step be of the form $\mathcal{M}_t := \mathcal{M}_{t-1} \cup C$. We prefer to incorporate erasure into the inference rule by allowing \mathcal{M}_t to be a *subset* of $\mathcal{M}_{t-1} \cup C$. One advantage of our definition is that the minimal length of a space-proof of \mathcal{T} , measured as the number of configurations in it, is precisely $\mathbf{Min} - \mathbf{Size}(\mathcal{T})$.

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 only clause that is not an immediate successor 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 $\mathcal{M}_{t-1} \rightsquigarrow \mathcal{M}_t$.

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. It can be readily verified that the clause space of a CNF \mathcal{T} , defined earlier, equals the minimal clause space of a space-proof of \mathcal{T} . To see this notice that every pebbling sequence of a derivation graph G_π that uses m pebbles gives rise to a space-proof with clause space m . In the other direction, given a space-proof with clause space m we can inductively construct a derivation π along with a derivation graph G_π associated with π that has pebbling cost m . To do so we introduce a new vertex v_t for each clause that is not an immediate successor of a clause in \mathcal{M}_{t-1} . If the t th step is an axiom download we make v_t a source. Otherwise the t th step is an inference, so we connect v_t to the vertices corresponding to its assumptions. Thus, from here on use $\mathbf{Min} - \mathbf{CSpace}(\mathcal{T})$ to denote the minimal clause space of a space-proof of π and the variable space of \mathcal{T} , denoted $\mathbf{Min} - \mathbf{VSpace}(\mathcal{T})$, is analogously defined.

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

Theorem 2.6. [ET01] For π any tree-like proof having size S , $\mathbf{CSpace}(\pi) \leq 1 + \log S$.

2.5 Restrictions

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

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

For $S = \{C_1, \dots\}$ a set or sequence of clauses let $S|_{x=a} = \{C_1|_{x=a}, \dots\}$ be the set or sequence obtained by restricting each member of it. We point out that if π is a resolution proof from \mathcal{T} and one removes from $\pi|_{x=a}$ all instances of 1, then one obtains a resolution proof of $\mathcal{T}|_{x=a}$. The proof obtained in this way may need to use the weakening rule. But we can further simplify the proof to eliminate use of weakening without increasing proof length, space or width. Thus, without loss of generality we shall assume $\pi|_{x=a}$ does not include any instances of 1.

2.6 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 2.7 (Decision Trees for 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 [BSIW04].

Lemma 2.8. For \mathcal{T} an unsatisfiable CNF, $\text{Min} - \text{TSize}(\mathcal{T}) = S_D(\mathcal{T})$

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 3.9](#), 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.
2. $\text{Min} - \text{TSize}(\mathcal{T}_n) = O(n)$, and $\text{Min} - \text{CSpace}(\mathcal{T}_n) = O(1)$. Moreover, there exist treelike proofs of \mathcal{T}_n with linear size and constant space simultaneously.
3. $\text{Min} - \text{Width}(\mathcal{T}_n) = O(1)$.

4. For any treelike proof π of \mathcal{T}_n :

$$\mathbf{Width}(\pi) \cdot \mathbf{CSpace}(\pi) \geq \Omega\left(\frac{n}{\log n}\right).$$

Consequently,

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

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 [BSW01].

Definition 3.2 (Pebbling Contradictions). Let G be a circuit, and $k > 0$ be an integer. Associate k distinct Boolean variables $x(v)_1, \dots, x(v)_k$ 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: $\bar{x}(u)_i \vee \bar{x}(w)_j \vee x(v)_1 \vee \dots \vee x(v)_k$ 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 3.10](#). 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. 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 3.3. *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 [Equation \(1\)](#) 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 [Equation \(1\)](#). Moreover, there is a derivation of [Equation \(1\)](#) from $\bigvee_{i=1}^k x(u)_i$ and $\bigvee_{i=1}^k x(w)_i$ and the Pebbling axioms for v along the following lines. For $j = 1, \dots, k$ derive $\bar{x}(w)_j \vee \bigvee_{i=1}^k x(v)_i$ from $\bigvee_{i=1}^k x(u)_i$ and the pebbling axioms $\{\bar{x}(u)_i \vee \bar{x}(w)_j \vee x(v)_1 \vee \dots \vee x(v)_k \mid i = 1, \dots, k\}$. For each j this requires k resolution steps and the maximal width of a clause in the process is $\max\{2k, k + 2\}$. Next, resolve $\bigvee_{i=1}^k x(w)_i$ with $\{\bar{x}(w)_j \vee \bigvee_{i=1}^k x(v)_i \mid j \in \{1, \dots, k\}\}$ to derive [Equation \(1\)](#). This requires k resolution steps and width at most $k + 1$, so the total number of steps needed to derive [Equation \(1\)](#) is $k(k + 1)$. Starting from the source axioms, one can infer [Equation \(1\)](#) for a sink $v \in T$ and then, resolving with the proper sink axioms, derive 0. \square

The following lemma proves [part 2](#) of [Theorem 3.1](#).

Lemma 3.4. *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 2.8) 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 3.5. *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. \square

Let us get back to proving Lemma 3.4. By Claim 3.5 each node in the tree has one branch that leads immediately to a falsifying assignment. 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. The resulting decision tree looks like a path of length $|V|$ with subtrees of size 1 protruding from each node. We conclude the tree, and the resolution proof that is equivalent to it, has linear size and can be pebbled with constant space. The lemma is proved. \square

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 3.9, we will need some technical introductory insights about the structure of proofs.

Definition 3.6 (Essential proofs). For π a proof of a CNF \mathcal{C} , and $\{\mathcal{M}_1 \dots \mathcal{M}_\ell\}$ 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 $\mathbf{0}$ appears in a memory configuration. Then $\mathbf{0}$ 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.

Observation 3.7. *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 this observation we can assume from now on that all our proofs are essential. Next, we list two 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} .

Observation 3.8. For $\pi = \{\mathcal{M}_0, \dots, \mathcal{M}_\ell\}$ an essential space-proof of \mathcal{C} :

1. ℓ is the first time $\mathbf{0}$ appears in a memory configuration, and $\mathcal{M}_\ell = \{\mathbf{0}\}$.
2. If x is the variable resolved upon in the inference step $\mathcal{M}_{t-1} \rightsquigarrow \mathcal{M}_t$, then

$$\text{Vars}(\mathcal{M}_{t-1}) - \{x\} \subseteq \text{Vars}(\mathcal{M}_t) \subseteq \text{Vars}(\mathcal{M}_{t-1}).$$

3.2 Proof of Part 4 of Theorem 3.1

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

Theorem 3.9. For any circuit G we have $\text{Min} - \text{VSpace}(Peb_G^1) \geq \text{BW} - \text{Peb}(G)$.

Before presenting the proof, let us demonstrate how this theorem implies part 4 of Theorem 3.1. The following theorem, due to [GT78] shows that there exist explicit constructions of graphs with large black-white pebbling measure.

Theorem 3.10. [GT78] For arbitrarily large n , there exist circuits G_n of size n such that

$$\text{BW} - \text{Peb}(G_n) = \Omega\left(\frac{n}{\log n}\right).$$

Let G be a graph with the properties stated in Theorem 3.10. Theorem 3.9 implies the first inequality in Part 4 because $\text{VSpace}(\pi) \leq \text{Width}(\pi) \cdot \text{CSpace}(\pi)$. By Theorem 2.6, if Peb_G^1 has a treelike proof of size S then this proof has clause space at most $\log S$. By Theorem 3.9 such a proof must have width at least $\frac{n}{\log S \cdot \log n}$. This proves the second inequality of Part 4. We now proceed to a proof of Theorem 3.9.

Proof: We claim that any essential proof of Peb_G^1 , 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_G^1 . For \mathcal{M} a set of clauses, define $\text{Positive}(\mathcal{M}) \subseteq \text{Vars}(\mathcal{M})$ 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 .

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

By its definition, the sequence $\{\mathcal{C}_0 \dots \mathcal{C}_\ell\}$ uses no more pebbles than $\text{VSpace}(\pi)$, 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 Observation 3.8, the only change in the set of literals in the t 'th step, would be focused on x_v . We know that $v \in \text{Black}(\mathcal{C}_{t-1})$ because $x_v \in \text{Positive}(\mathcal{M}_{t-1})$. Let us examine the possible cases. If $x_v \notin \text{Vars}(\mathcal{M}_t)$ we can remove the black pebble from v . If $x_v \in \text{Positive}(\mathcal{M}_t)$, then we leave the black pebble on v at time t . Finally, if $x_v \in \text{Vars}(\mathcal{M}_t) - \text{Positive}(\mathcal{M}_t)$, 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 black or 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 a 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 $\{\mathcal{C}_0 \dots \mathcal{C}_\ell\}$ are legal. By the soundness of resolution, any proof of Peb_G^1 must use the sink axiom \bar{x}_s , because $Peb_G^1 - \{\bar{x}_t\}$ 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 $\{\mathcal{C}_0 \dots \mathcal{C}_\ell\}$ is a legal BW-pebbling of G . The theorem is proved. \square

4 On Size and Space in 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 cannot have simultaneously small size and small space. The method of replacing each variable by two variables was exploited in [BSW01] in order to obtain an exponential separation of treelike and general resolution. The form we use here was first introduced by Alekhovich and Razborov [AR01], 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} . Applying this transformation to our pebbling formula from the previous section, we will get our results. As mentioned in the introduction, the main result of this section, [Theorem 4.3](#), has been subsumed recently by [BSN08] which proved for every graph G that

$$\text{Min} - \text{CSpace}(\mathcal{X}(Peb_G^1)) \geq \text{BW} - \text{Peb}(G).$$

We include the proof because its techniques may have use in other inquiries into resolution size and space.

Definition 4.1 (Xorification). For $\mathcal{T}(x_1, \dots, x_n)$ a CNF, we define its *xorification* to be the following formula over variables $\{x'_1, x''_1, x'_2, x''_2, \dots, x'_n, x''_n\}$. The *xorification* of a literal x_i^a , $a \in \{0, 1\}$, is the formula

$$\mathcal{X}(x_i^a) \stackrel{\text{def}}{=} x'_i \oplus x''_i \oplus a.$$

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. More to the point, 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.

Following [AR01], we apply a random restriction to the xorification of a pebbling contradiction and derive the following result for general resolution.

Theorem 4.2. *For any CNF \mathcal{T} and any resolution proof π of $\mathcal{X}(\mathcal{T})$,*

$$\text{CSpace}(\pi) \cdot \log_{\frac{4}{3}} \text{Size}(\pi) \geq \text{Min} - \text{VSpace}(\mathcal{T}).$$

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 $\frac{1}{2}$. Notice $\mathcal{X}(\mathcal{T})|_\rho$ is equivalent to \mathcal{T} up to renaming of variables and literals. Look at a clause $C \in \pi$. We claim that

$$\Pr[C|_\rho \neq 1] \leq \left(\frac{3}{4}\right)^{|C|}.$$

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, at most

$$\mathbf{Size}(\pi) \cdot \left(\frac{3}{4}\right)^w.$$

Set $w = \frac{\mathbf{Min-VSspace}(\mathcal{T})}{\mathbf{CSpace}(\pi)}$ and conclude that if $\mathbf{Size}(\pi) < (\frac{4}{3})^w$, then there exists a restriction ρ for which $\pi|_\rho$ has width w . Notice that clause space does not increase under restrictions, hence

$$\mathbf{Width}(\pi|_\rho) \cdot \mathbf{CSpace}(\pi|_\rho) < \mathbf{Min-VSspace}(\mathcal{T}),$$

which contradicts the fact that $\mathbf{Width}(\pi|_\rho) \cdot \mathbf{CSpace}(\pi|_\rho) \geq \mathbf{Min-VSspace}(\mathcal{T})$. We conclude

$$\log_{\frac{4}{3}} \mathbf{Size}(\pi) \geq \frac{\mathbf{Min-VSspace}(\mathcal{T})}{\mathbf{CSpace}(\pi)},$$

and the theorem is proved. \square

We can now state our main result regarding simultaneous size-space efficient proofs 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 simultaneously.
2. $\mathbf{Min-VSspace}(\mathcal{T}_n) = \Omega(\frac{n}{\log n})$.
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 $\mathcal{T}_n = \mathcal{X}(Peb_G^1)$ for G a graph over n vertices with black-white pebbling measure of $\Omega(\frac{n}{\log n})$. Part 1 is immediate from the proof of [Lemma 3.3](#), 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_G^1 . Since a restriction does not increase the variable space, this proves part 2. Finally, part 3 follows from [Theorem 4.2](#). \square

5 Discussion and Applications

5.1 Finding Minimal Width Proofs

The size-width relations of [\[BSW01\]](#) 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 short proof to a small-width proof. In this process, as it was presented in [\[BSW01\]](#), 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 short proof of Peb_G^1 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 2.6](#)).

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 [BSW01] 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_G^1 , then necessarily the clause space will be large, and hence also the treelike size.

5.3 Other SAT Solving Heuristics

Some of the most extensively 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 [DLL62], 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_G^1 follows this heuristic. See proof of Lemma 3.4).

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_G^1 , 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. $\text{Min} - \text{TSize}(\mathcal{T}_n) = O(n)$.
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 where each layer has 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_G^1 , and replace each source clause $x_{v_1^b}$ (for $b \in \{1, 2\}$) by (say) the CNF

$$\bigwedge_{(\epsilon_1, \dots, \epsilon_4) \in \{0,1\}^4} \left(\left(\bigvee_{i=1}^4 y_i^{\epsilon_i} \right) \vee x_{v_1^b} \right)$$

(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 5.2. $\text{Min} - \text{TSize}(\mathcal{T}_n) = O(n)$

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 3.4](#), has a small decision tree. The claim is proved. \square

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

Proof: By induction on n . For $n = 0$, $\mathcal{T}_0|_{x_s=0} = \mathbf{0}$, 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. \square

Acknowledgements

We thank Nicola Galesi and Jakob Nordström for helpful discussions. We thank the anonymous referees for comments that greatly improved the clarity of the paper.

References

- [ABSRW02] Michael Alekhovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002.
- [AJPU02] Michael Alekhovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC-02)*, pages 448–456, New York, May 19–21 2002. ACM Press.
- [AR01] Michael Alekhovich and Alexander Razborov. Personal communication, 2001.
- [BEGJ00] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM J. Comput.*, 30(5):1462–1484, 2000.
- [BKPS02] Paul Beame, Richard Karp, Toniann Pitassi, and Michael Saks. The efficiency of resolution and davis-putnam procedures. *SIAM J. Comput.*, 31(4):1048–1075, 2002.
- [BL78] P. Emde Boas and J. van Leeuwen. Move-rules and trade-offs in the pebble game. Technical Report RUU-CS-78-04, Department of Information and Computing Sciences, Utrecht University, 1978.
- [Bla37] Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, University of Chicago, 1937.
- [BOMP01] Josh Buresh-Oppenheim, David Mitchell, and Toniann Pitassi. Linear and negative resolution are weaker than resolution. Technical Report TR01-074, Electronic Colloquium on Computational Complexity (ECCC), 2001. Available at <http://www.eccc.uni-trier.de/eccc/>.

- [BS02] Eli Ben-Sasson. Size space tradeoffs for resolution. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pages 457–464, May 2002.
- [BSG03] Eli Ben-Sasson and Nicola Galesi. Space complexity of random formulae in resolution. *Random Struct. Algorithms*, 23(1):92–109, 2003.
- [BSIW04] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of treelike and general resolution. *Combinatorica*, 24(4):585–603, September 2004.
- [BSN08] Eli Ben-Sasson and Jakob Nordström. Short proofs may be spacious: An optimal separation of space and length in resolution (*submitted*), October 2008.
- [BSW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001.
- [CS76] Stephen A. Cook and Ravi Sethi. Storage requirements for deterministic polynomial time recognizable languages. *Journal of Computer and System Sciences*, 13(1):25–37, 1976.
- [CS88] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.
- [DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.
- [DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.
- [ET01] Juan Luis Esteban and Jacobo Torn. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001.
- [Gal77] Zvi Galil. On resolution with clauses of bounded size. *SIAM Journal on Computing*, 6(3):444–459, 1977.
- [GT78] John R. Gilbert and Robert Endre Tarjan. Variations of a pebble game on graphs. Technical Report STAN-CS-78-661, Stanford University, 1978. Available at the webpage <http://infolab.stanford.edu/TR/CS-TR-78-661.html>.
- [Hak85] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.
- [HP07] Philipp Hertel and Toniann Pitassi. Exponential time/space speedups for resolution and the PSPACE-completeness of black-white pebbling. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*, pages 137–149, October 2007.
- [HPV77] John Hopcroft, Wolfgang Paul, and Leslie Valiant. On time versus space. *Journal of the ACM*, 24(2):332–337, April 1977.
- [NH08] Jakob Nordström and Johan Håstad. Towards an optimal separation of space and length in resolution. In Richard E. Ladner and Cynthia Dwork, editors, *STOC*, pages 701–710. ACM, 2008.

- [Nor06] Jakob Nordström. Narrow proofs may be spacious: Separating space and width in resolution (Extended abstract). In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC '06)*, pages 507–516, May 2006. Journal version to appear in *SIAM Journal on Computing*.
- [Nor09] Jakob Nordström. New wine into old wineskins: A survey of some pebbling classics with supplemental results. Manuscript in preparation, 2009.
- [Pip80] Nicholas Pippenger. Pebbling. Technical Report RC8258, IBM Watson Research Center, 1980. Appeared in Proceedings of the 5th IBM Symposium on Mathematical Foundations of Computer Science, Japan.
- [PR04] Toniann Pitassi and Ran Raz. Regular resolution lower bounds for the weak pigeonhole principle. *Combinatorica*, 24(3):503–524, 2004.
- [Raz04] Ran Raz. Resolution lower bounds for the weak pigeonhole principle. *Journal of the ACM*, 51(2):115–138, March 2004.
- [RM99] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, March 1999.
- [Rob65] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.
- [Tor99] Jacobo Torán. Lower bounds for space in resolution. In *Proceedings of the 13th International Workshop on Computer Science Logic (CSL '99)*, volume 1683 of *Lecture Notes in Computer Science*, pages 362–373. Springer, 1999.
- [Tse68] Grigori Tseitin. On the complexity of derivation in propositional calculus. In A. O. Silenko, editor, *Structures in Constructive Mathematics and Mathematical Logic, Part II*, pages 115–125. Consultants Bureau, New York-London, 1968.
- [Urq87] Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, January 1987.
- [WS72] S. A. Walker and H. R. Strong. Characterizations of flowchartable recursions short version. In *STOC '72: Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 18–34, New York, NY, USA, 1972. ACM.