Lower Bounds for Width-Restricted Clause Learning on Formulas of Small Width

Eli Ben-Sasson
Technion – Israel Institute of Technology
Haifa, Israel
eli@cs.technion.ac.il

Jan Johannsen
LMU München
Munich, Germany
jan.johannsen@ifi.lmu.de

Abstract

Clause learning is a technique used by backtracking-based propositional satisfiability solvers, where some clauses obtained by analysis of conflicts are added to the formula during backtracking. It has been observed empirically that clause learning does not significantly improve the performance of a solver when restricted to learning clauses of small width only. This experience is supported by lower bound theorems. It is shown that lower bounds on the runtime of width-restricted clause learning follow from lower bounds on the width of resolution proofs. This yields the first lower bounds on width-restricted clause learning for formulas in 3-CNF.

1 Introduction

In the past decades, a considerable amount of research has been devoted to the satisfiability problem for classical propositional logic (SAT). Besides its central role in computational complexity theory, programs for this problem, so-called SAT solvers, are of increasing importance for practical applications in various domains.

Most contemporary SAT solvers are based on extensions of the backtracking procedure known as the DLL algorithm [Davis *et al.*, 1962]. Commonly used extensions to the basic DLL algorithm described below include *non-chronological backtracking* [Bayardo Jr. and Schrag, 1997], *clause learning* [Marques-Silva and Sakallah, 1996] and *restarts* [Gomes *et al.*, 1997].

The basic recursive DLL procedure is called for a formula F in conjunctive normal form and a partial assignment α (which is empty in the initial call from outside). If α satisfies F, then α is returned, and if α causes a conflict, i.e., falsifies a clause in F, then the call fails. Otherwise a variable x that occurs in $F\lceil\alpha$, the formula remaining after α is applied to F, is selected according to some heuristic, and the procedure is called recursively twice, once with α extended by $x := 1$ and once with α extended by $x := 0$. If one of the recursive calls returns a satisfying assignment, then this assignment is returned, otherwise – if both recursive calls fail – the call fails.

An important ingredient in implementations of the DLL algorithm is *unit propagation*. Abstractly, it can be seen as just a variable selection heuristic: if there is a variable x that occurs in a unit clause in $F\lceil\alpha$, i.e., a clause containing just one unevaluated literal, then this variable x is selected. Since one of the two recursive calls then leads to a conflict immediately, the assignment α can instead be extended directly by setting x so that the literal in the unit clause is satisfied. This is iterated until no unit clauses remain, thereby closing α under implications. Note that conflicts are always detected during unit propagation, since immediately before being falsified a clause will be a unit clause.

One of the most successful extensions of the DLL algorithm is *clause learning* [Marques-Silva and Sakallah, 1996]: when the procedure encounters a conflict, then a sub-assignment α' of α is computed that suffices to cause this conflict, i.e., such that the closure of α' under unit propagation already falsifies F. This sub-assignment α' can then be stored in form of a new clause C added to the formula, viz. the unique largest clause C falsified by α'. This way, when in a later branch of the search another partial assignment extending α' occurs, the procedure can backtrack earlier since then the added clause C becomes falsified and causes a conflict.

When clause learning is implemented, a heuristic, the *learning strategy*, is needed to decide which learnable clauses to keep in memory, because learning a large number of clauses leads to excessive consumption of memory, which slows the solver down rather than helping it. Many early learning strategies were such that the *width*, i.e., the number of literals, of learnable clauses was restricted, so that the solver learned only clauses whose width does not exceed a certain threshold.

Experience has shown that such learning strategies are not very helpful, i.e., learning only narrow clauses does not significantly improve the performance of a DLL algorithm for hard formulas. The present paper continues a line of work that aims at supporting this experience with rigorous mathematical analyses in the form of lower bound theorems.

The first lower bound for width-restricted clause learning was shown [Buss *et al.*, 2008] for the well-known pigeonhole principle clauses PHP_n. These formulas require time

The work of the first author was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) unter grant agreement number 240258.

$2^{\Omega(n \log n)}$ to solve when learning clauses of width up to $n/2$ only, whereas they can be solved in time $2^{O(n)}$ when learning arbitrary clauses. While this example in principle shows that learning wide clauses can yield a speed-up, it is not fully satisfactory, since even with arbitrary learning, the time required is exponential in n.

Another lower bound was shown [Johannsen, 2009] for a a set of clauses Ord_n based on the ordering principle, which states that every partial ordering on n elements has a maximal element. These formulas can be solved in polynomial time when learning arbitrary clauses, but require exponential time to solve when learning clauses of size up to $n/4$ only.

Both lower bounds are asymptotically the same as the known lower bounds [Iwama and Miyazaki, 1999; Bonet and Galesi, 2001] on the time required for solving the respective formulas by DLL algorithms without clause learning.

In these previous lower bounds, the hard example formulas PHP_n and Ord_n themselves contain clauses of large width. Since it is conceivable that the necessity to learn wide clauses is merely due to the presence of these wide initial clauses, the question arose whether similar lower bounds can be shown for formulas of small width. We answer this question by proving lower bounds on width-restricted clause learning for small width formulas.

The lower bounds are shown by proving the same lower bounds on the size of refutations in a certain resolution based propositional proof system RTL (see Section 2). The relationship of this proof system to the DLL algorithm with clause learning has been established in several earlier works [Buss *et al.*, 2008; Hertel *et al.*, 2008]. We will show that for formulas of small width, lower bounds for this proof system follow from lower bounds on the width of resolution proofs. This also gives an easier proof of a slightly weaker form of the previous lower bound [Johannsen, 2009] for the formulas Ord_n.

Using known constructions [Bonet and Galesi, 2001; Segerlind *et al.*, 2004] of families of unsatisfiable 3-CNF formulas that have resolution refutations of polynomial size, but which require resolution refutations of large width, we obtain a number of results which show that width-restricted clause learning algorithms will require exponentially longer running time than clause learning algorithms with unrestricted width, for certain hard formulas having small width.

The lower bound for clause learning algorithms on formulas requiring large resolution width is somewhat dual to a result of Atserias et al. [2009], who give a small polynomial upper bound on the runtime of a clause learning algorithm *with restarts* on formulas having resolution refutations of small width.

We will now briefly describe the idea for our lower bound proof. In the proof of the lower bound for the pigeonhole principle formulas PHP_n [Buss *et al.*, 2008], it was shown that it takes a long time to derive clauses that are small enough to be learned. This is not the case for the clauses Ord_n, from which small clauses can be derived very quickly. In the lower bound proof for these formulas [Johannsen, 2009], the clauses were semantically classified into *useful* and *useless* clauses, and it was shown that on the one hand, learning useless clauses does not reduce the running time significantly, and on the other hand it takes a long time to derive sufficiently small useful clauses.

In this work, the main idea for the lower bound proof is similar to that for the clauses Ord_n. The classification of clauses into useful and useless clauses in this general case is obtained from a combinatorial characterization of formulas requiring resolution refutations of large width, due to Atserias and Dalmau [2008].

2 Preliminaries

A *literal* a is a variable $a = x$ or a negated variable $a = \bar{x}$. A *clause* C is a disjunction $C = a_1 \vee \ldots \vee a_k$ of literals a_i. The *width* of C is k, the number of literals in C.

A formula in *conjunctive normal form* (CNF) is a conjunction $F = C_1 \wedge \ldots \wedge C_m$ of clauses, it is usually identified with the set of clauses $\{C_1, \ldots, C_m\}$. A formula F in CNF is in k-CNF if every clause C in F is of width $w(C) \leq k$.

We consider resolution-based refutation systems for formulas in CNF, which are known to be strongly related to DLL algorithms. These proof systems have two inference rules: the *weakening rule*, which allows to conclude a clause D from any clause C with $C \subseteq D$, and the *resolution rule*, which allows to infer the clause $C \vee D$ from the two clauses $C \vee x$ and $D \vee \bar{x}$, provided that the variable x does not occur in either C or D, pictorially:

$$\frac{C \vee x \qquad D \vee \bar{x}}{C \vee D}$$

We say that the variable x is *eliminated* in this inference.

A resolution derivation of a clause C from a CNF-formula F is a directed acyclic graph (dag) with a unique sink, in which every node has in-degree at most 2, where every node v is labeled with a clause C_v such that:

1. The sink is labeled with C.
2. If a node v has one predecessor u, then C_v follows from C_u by the weakening rule.
3. If a node v has two predecessors u_1, u_2, then C_v follows from C_{u_1} and C_{u_2} by the resolution rule.
4. A source node ν is labeled by a clause C in F.

A *resolution refutation* of F is a resolution derivation of the empty clause from F. Resolution is sound and complete: a CNF-formula F has a resolution refutation if and only if it is unsatisfiable.

We call a derivation *tree-like* if the underlying unlabeled dag is a tree, otherwise we may call it *dag-like* for emphasis. As usual, for a dag that is a tree we refer to the sink as the *root*, to the predecessors of a node as its *children* and to a source node as a *leaf*.

The *size* of a resolution derivation is the number of nodes in the dag. The *width* of a resolution refutation R is the maximal width of a clause occurring in R. The *resolution width* of F is the minimal width of a resolution refutation of F.

Ben-Sasson and Wigderson [2001] have shown the following relation between resolution width and size of tree-like resolution:

Theorem 1. *If a d-CNF formula F requires resolution width at least w, then every tree-like resolution refutation of F is of size at least 2^{w-d}.*

In the literature, resolution proof systems are sometimes defined without the weakening rule, but since applications of this rule can be eliminated from a tree-like resolution refutation without increasing the size or width, all lower bounds shown for tree-like resolution without weakening apply to the system with weakening as well.

Let X be a set of variables. A *restriction* ρ of X is a partial assignment $X \to \{0,1\}$. A restriction ρ is extended to literals by setting

$$\rho(\bar{x}) := \begin{cases} 1 & \text{if } \rho(x) = 0 \\ 0 & \text{if } \rho(x) = 1 \end{cases}$$

For a clause C in variables X, we define

$$C\lceil\rho := \begin{cases} 1 & \text{if } \rho(a) = 1 \text{ for some } a \in C \\ \bigvee\limits_{a \in C,\, \rho(a) \neq 0} a & \text{otherwise,} \end{cases}$$

where the empty disjunction is identified with the constant 0. For a CNF-formula F over X, we define

$$F\lceil\rho := \begin{cases} 0 & \text{if } C\lceil\rho = 0 \text{ for some } C \in F \\ \bigwedge\limits_{C \in F,\, C\lceil\rho \neq 1} C\lceil\rho & \text{otherwise,} \end{cases}$$

where the empty conjunction is identified with 1.

Proposition 2. *Let R be a (tree-like) resolution derivation of C from F of size s, and ρ a restriction. Then there is a (tree-like) resolution derivation R' of $C\lceil\rho$ from $F\lceil\rho$ of size at most s.*

In particular, if $C\lceil\rho = 0$ then R' is a resolution refutation of $F\lceil\rho$. As usual, we denote the derivation R' by $R\lceil\rho$.

A resolution derivation is called *regular* if on every path through the dag, no variable is eliminated twice. This condition is inessential for tree-like resolution since minimal tree-like refutations are always regular [Tseitin, 1968], but regular dag-like refutations can necessarily be exponentially longer than general ones [Alekhnovich *et al.*, 2007].

Tree-like resolution exactly corresponds to the DLL algorithm by the following well-known correspondence: the search tree produced by the run of a DLL algorithm on an unsatisfiable formula F forms a tree-like resolution refutation of F, and from a given tree-like regular resolution refutation of F one can construct a run of a DLL algorithm showing the unsatisfiability of F that produces essentially the given search tree.

In order to define proof systems that correspond to the DLL algorithm with clause learning in the same way, we define *resolution trees with lemmas* (RTL). In these proof systems, the order of branches in the proof tree is significant, thus the underlying trees need to be ordered.

An *ordered binary tree* is a rooted tree in which every node has at most 2 children, and where every node with 2 children has a distinguished *left* and *right* child. The post-ordering $\prec$ of an ordered binary tree is the order in which the nodes of the tree are visited by a post-order traversal, i.e., $u \prec v$ holds for nodes u, v if u is a descendant of v, or if there is a common ancestor w of u and v such that u is a descendant of the left child of w and v is a descendant of the right child of w.

An RTL-derivation of a clause C from a CNF-formula F is an ordered binary tree, in which every node v is labeled with a clause C_v such that:

1. The root is labeled with C.
2. If a node v has one child u, then C_v follows from C_u by the weakening rule.
3. If a node v has two children u_1, u_2, then C_v follows from C_{u_1} and C_{u_2} by the resolution rule.
4. A leaf v is labeled by a clause D in F, or by a clause C labeling some node $u \prec v$. In the latter case we call C a *lemma*.

An RTL-derivation is an $\mathrm{RTL}(k)$-derivation if every lemma C is of width $w(C) \leq k$. An RTL-refutation of F is an RTL-derivation of the empty clause from F.

A subsystem WRTI of RTL was defined by Buss et al. [2008], which exactly corresponds to a general formulation of the DLL algorithm with clause learning: the size of a refutation of an unsatisfiable formula F in WRTI has been shown [Buss *et al.*, 2008] to be polynomially related to the runtime of a schematic algorithm DLL-L-UP on F. This schema DLL-L-UP subsumes most clause learning strategies commonly used in practice, including *first-UIP* [Marques-Silva and Sakallah, 1996], *all-UIP*, *decision* [Zhang *et al.*, 2001] and *rel-sat* [Bayardo Jr. and Schrag, 1997]. A variant of DLL-L-UP which incorporates these learning strategies and also allows for non-chronological backtracking [Bayardo Jr. and Schrag, 1997] was described by Hoffmann [2007] and shown to be likewise simulated by WRTI.

It follows from the mentioned results of Buss et al. [2008] that if an unsatisfiable formula F can be solved by a DLL algorithm with clause learning in time t, then it has an RTL-refutation of size polynomial in t. Moreover, if the algorithm learns only clauses of width at most k, then the refutation is in $\mathrm{RTL}(k)$. In this work we prove lower bounds on the size of $\mathrm{RTL}(k)$-refutations, which thus yield lower bounds on the runtime of DLL algorithms with width-restricted clause-learning.

In addition to clause learning, most state-of-the-art satisfiability solvers also use *restarts* [Gomes *et al.*, 1997], where the search is periodically discarded and started from scratch, retaining only the learned clauses. The performance of such solvers is thus not modelled by RTL. The runtime of a DLL algorithm with clause learning and restarts was shown to be polynomially related to the size of general dag-like resolution refutations, for certain particular learning strategies [Beame *et al.*, 2004] and more recently also for most natural learning strategies [Pipatsrisawat and Darwiche, 2009]. However, these simulations of general dag-like resolution proofs, as well as the clause learning algorithm of Atserias et al. [2009] that simulates resolution proofs of small width, use a particular restart policy: they perform a restart after every conflict. An interesting question is whether general resolution proofs can be simulated with more natural restart policies.

3 Resolution Width and Systems of Restrictions

Let X be a set of variables, and $w \in \mathbb{N}$ with $w \leq |X|$. A *w-system of restrictions* over X is a non-empty set $\mathcal{H}$ of restrictions with the following properties:

- $|\rho| \leq w$ for all $\rho \in \mathcal{H}$,
- downward closure: if $\rho \in \mathcal{H}$ and $\rho' \subseteq \rho$, then $\rho' \in \mathcal{H}$,
- the extension property: if $\rho \in \mathcal{H}$ with $|\rho| < w$, and $x \in X \setminus \operatorname{dom}\rho$, then there is $\rho' \in \mathcal{H}$ with $\rho' \supseteq \rho$ and $x \in \operatorname{dom}\rho'$.

We say that $\mathcal{H}$ *avoids* a clause C if $C\restriction\rho \neq 0$ for every $\rho \in \mathcal{H}$, and $\mathcal{H}$ avoids a formula F if $\mathcal{H}$ avoids every clause $C \in F$.

The notion was introduced by Atserias and Dalmau [2008], who showed the following characterization of resolution width:

Theorem 3. *A formula F requires resolution width at least w if and only if there is a w-system of restrictions over $var(F)$ that avoids F.*

Atserias and Dalmau [2008] called a w-system of restrictions avoiding F a *winning strategy for the Duplicator in the Boolean existential w-pebble game on F*, which is explained by the origin of the notion in the existential k-pebble game [Kolaitis and Vardi, 1995] in finite model theory. Since we make no use of the model-theoretic background, we chose to use a shorter name for the concept.

For our application we shall use the concept of a system of restrictions being restricted by one of its elements, which we define now.

Lemma 4. *If $\mathcal{H}$ is a w-system of restrictions over X, and $\rho \in \mathcal{H}$ with $|\rho| = r < w$, then the set*

$$\mathcal{H}\restriction\rho := \{\, \sigma \,;\, \operatorname{dom}\sigma \subseteq X \setminus \operatorname{dom}\rho \text{ and } \sigma \cup \rho \in \mathcal{H} \text{ and } |\sigma| \leq w - r \,\}$$

is a $(w-r)$-system of restrictions over $X \setminus \operatorname{dom}\rho$.

Note that $\mathcal{H}\restriction\rho$ would be empty, and hence not a system of restrictions in the sense of the definition, if the definition were extended to restrictions $\rho \notin \mathcal{H}$: if there is a $\sigma \in \mathcal{H}\restriction\rho$, then by definition $\sigma \cup \rho \in \mathcal{H}$, and by downward closure $\rho \in \mathcal{H}$.

Proof. Every $\sigma \in \mathcal{H}\restriction\rho$ has $|\sigma| \leq w - r$ by definition. If $\sigma \in \mathcal{H}\restriction\rho$ and $\sigma' \subseteq \sigma$, then $\sigma' \cup \rho \subseteq \sigma \cup \rho$, and thus by downward closure of $\mathcal{H}$ we have $\sigma' \cup \rho \in \mathcal{H}$. Therefore $\sigma' \in \mathcal{H}\restriction\rho$, hence $\mathcal{H}\restriction\rho$ is downward closed.

If $\sigma \in \mathcal{H}\restriction\rho$ is a restriction with $|\sigma| < w - r$ and $x \in X \setminus \operatorname{dom}\rho$ is a variable with $x \notin \operatorname{dom}\sigma$, then $|\sigma \cup \rho| < w$, and hence by the extension property of $\mathcal{H}$ there is $\sigma' \supseteq \sigma \cup \rho$ in $\mathcal{H}$ with $x \in \operatorname{dom}\sigma'$. Then $\sigma' \setminus \rho \supseteq \sigma$ is in $\mathcal{H}\restriction\rho$, and $x \in \operatorname{dom}(\sigma' \setminus \rho)$. Therefore $\mathcal{H}\restriction\rho$ has the extension property, and hence is a $(w-r)$-system of restrictions over $X \setminus \operatorname{dom}\rho$. □

Lemma 5. *If $\mathcal{H}$ is a w-system of restrictions that avoids F, and $\rho \in \mathcal{H}$, then $\mathcal{H}\restriction\rho$ avoids $F\restriction\rho$.*

Proof. Assume that $\mathcal{H}\restriction\rho$ does not avoid $F\restriction\rho$, i.e., there is a clause C in $F\restriction\rho$ and a restriction $\sigma \in \mathcal{H}\restriction\rho$ such that $C\restriction\sigma = 0$. Since C is in $F\restriction\rho$, there is a clause D with $D\restriction\rho = 0$ such that $C \vee D \in F$. By definition, $\sigma' = \sigma \cup \rho \in \mathcal{H}$ and $(C \vee D)\restriction\sigma' = C\restriction\sigma \vee D\restriction\rho = 0$, hence $\mathcal{H}$ does not avoid F, in contradiction to the hypothesis. □

4 The Lower Bound

We now prove our main theorem, which shows that lower bounds for $\mathrm{RTL}(k)$-refutations of F follow from lower bounds on the resolution width of F, for formulas F of sufficiently small width.

Theorem 6. *If F is a d-CNF that requires resolution width at least w to refute, then for any k, every $\mathrm{RTL}(k)$-refutation of F is of size at least*

$$2^{w-(k+\max\{d,k\})} \geq 2^{w-(2k+d)}.$$

Proof. Let R be an $\mathrm{RTL}(k)$-refutation of F. Since F requires resolution width w, by Theorem 3, there is a w-system of restrictions $\mathcal{H}$ that avoids F.

Let C be the first clause in R that is of small enough width $w(C) \leq k$ to be used as a lemma, and that is not avoided by $\mathcal{H}$. In particular, every lemma in R derived before C *is* avoided by $\mathcal{H}$. Let ρ be the smallest restriction in $\mathcal{H}$ with $C\restriction\rho = 0$, so that we have $r := |\rho| = w(C) \leq k$.

Let R_C be the subtree of R below C, so that R_C is an $\mathrm{RTL}(k)$-derivation of C from F. Let G be the set of lemmas used in R_C, thus R_C is a tree-like resolution derivation of C from $F \wedge G$, and therefore $R' := R_C\restriction\rho$ is a tree-like resolution refutation of $F' := (F \wedge G)\restriction\rho$. Note that every clause in F is of width d, and every clause in G is of width k, therefore $w(F') \leq w(F \wedge G) \leq \max\{d,k\}$.

By the choice of C we know that $\mathcal{H}$ avoids every clause in G, and hence $\mathcal{H}$ avoids $F \wedge G$. It follows by Lemmas 4 and 5 that $\mathcal{H}\restriction\rho$ is a $(w-r)$-system of restrictions that avoids F'.

Therefore, by Theorem 3, F' requires resolution width $w - r \geq w - k$, and thus by Theorem 1, the refutation $R_C\restriction\rho$, and therefore R, is of size at least $2^{(w-k)-w(F')} \geq 2^{w-(k+\max\{d,k\})}$ as claimed. □

5 Applications

We now instantiate our general lower bound to prove several lower bounds for $\mathrm{RTL}(k)$-refutations of certain concrete formulas.

Ordering Principle

The ordering principle expresses the fact that every finite linear ordering has a maximal element. Its negation is expressed in propositional logic by the following set of clauses Ord_n over the variables $x_{i,j}$ for $1 \leq i, j \leq n$ with $i \neq j$:

$$\begin{array}{lll}
\bar{x}_{i,j} \vee \bar{x}_{j,i} & \text{for } 1 \leq i < j \leq n & (A_{i,j}) \\
x_{i,j} \vee x_{j,i} & \text{for } 1 \leq i < j \leq n & (T_{i,j}) \\
\bar{x}_{i,j} \vee \bar{x}_{j,k} \vee \bar{x}_{k,i} & \text{for } 1 \leq i,j,k \leq n \text{ pairwise distinct} & (\Delta_{i,j,k}) \\
\displaystyle\bigvee_{1 \leq j \leq n,\, j \neq i} x_{i,j} & \text{for } 1 \leq i \leq n & (M_i)
\end{array}$$

The clauses $A_{i,j}$, $T_{i,j}$ and $\Delta_{i,j,k}$ state that in a satisfying assignment, the values of the variables define a linear ordering

on n points. The clause M_i expresses that i is not a maximum in this ordering, therefore the formula Ord_n is unsatisfiable.

The formulas Ord_n were introduced by Krishnamurthy [1985] as potential hard example formulas for resolution, but short regular resolution refutations for them were constructed by Stålmarck [1996].

Proposition 7. *There are dag-like regular resolution refutations of* Ord_n *of size* $O(n^3)$.

Note that the size of the formula Ord_n is $\Theta(n^3)$, so the size of these refutations is linear in the size of the formula. A general simulation of regular resolution by WRTI [Buss *et al.*, 2008] yields WRTI-refutations of Ord_n of polynomial size. On the other hand, a lower bound for $\mathrm{RTL}(k)$-refutations of Ord_n was shown by Johannsen [2009]:

Theorem 8. *For* $k < n/4$*, every* $\mathrm{RTL}(k)$*-refutation of* Ord_n *is of size* $2^{\Omega(n)}$.

Thus this lower bound shows the necessity to use wide lemmas to refute them efficiently. But since the formula Ord_n itself contains wide clauses, it is conceivable that it is these wide clauses that cause this necessity. We therefore apply our general lower bound to derive similar lower bounds for variants of the ordering principle formulas having small width. The most straightforward way to obtain a formula of small width from any formula is to expand it into a 3-CNF, as described below:

For a CNF-formula F, the 3-CNF-expansion $E_3(F)$ of F is obtained as follows: for every clause $C = a_1 \vee \ldots \vee a_k$ in F of width $w(C) = k \geq 4$, introduce $k+1$ new *extension variables* $y_{C,0}, \ldots, y_{C,k}$, and replace C by the clauses:

$$y_{C,0} \qquad \bar{y}_{C,i-1} \vee a_i \vee y_{C,i} \quad \text{for } 1 \leq i \leq k \qquad \bar{y}_{C,k}$$

The formula $E_3(F)$ is obviously in 3-CNF and is satisfiable if and only if F is satisfiable.

Bonet and Galesi [2001] show a lower bound of $n/6$ on the resolution width of the 3-CNF expansion $E_3(\mathrm{Ord}_n)$ of the ordering principle. In fact, we can show the following slightly larger lower bound.

Theorem 9. *The formula* $E_3(Ord_n)$ *requires resolution width at least* $n/2$.

By Theorem 6, a lower bound for $\mathrm{RTL}(k)$-refutations of $E_3(\mathrm{Ord}_n)$ follows from Theorem 9: by choosing $k = n/6$ and observing that for $n \geq 18$ we get $k \geq 3$, we obtain from Theorem 6 a lower bound of $2^{n/2-2n/6} = 2^{n/6}$.

Corollary 10. *For* $n \geq 18$*, every* $\mathrm{RTL}(n/6)$*-refutation of* $E_3(\mathrm{Ord}_n)$ *is of size* $2^{n/6}$.

It follows that a DLL algorithm with clause learning requires exponential time to solve the formulas $E_3(\mathrm{Ord}_n)$ when only clauses of width $n/6$ are learned. On the other hand, from the short regular resolution refutations of Ord_n, short regular refutations of $E_3(\mathrm{Ord}_n)$ are obtained easily. From those, one can construct a run of a DLL algorithm with arbitrary clause learning on $E_3(\mathrm{Ord}_n)$ in polynomial time. Hence we have an example of 3-CNF formulas for which learning wide clauses is necessary to solve them efficiently.

Since the clauses M_i have tree-like derivations from $E_3(M_i)$ of size n, an $\mathrm{RTL}(k)$-refutation of Ord_n of size s can be converted into an $\mathrm{RTL}(k)$-refutation of $E_3(\mathrm{Ord}_n)$ of size sn. Hence Corollary 10 also yields an easier proof of a slightly weaker variant of the lower bound from Theorem 8: every $\mathrm{RTL}(n/6)$-refutation of Ord_n is of size at least $2^{n/6-\log n}$.

Graph Ordering Principle

A different way to obtain a small width formula from the ordering principle is to consider the restriction of it to a graph, as introduced by Segerlind et al. [2004]. The only wide clauses in Ord_n are the clauses M_i stating that there is an element larger than i, for every i. A formula of small width can be obtained by defining for every i a small set of elements and requiring that one element in this set is larger than i.

For a graph $G = (V, E)$ on n vertices $V = \{1, \ldots, n\}$, the formula $\mathrm{Ord}(G)$ consists of the clauses $A_{i,j}$, $T_{i,j}$ and $\Delta_{i,j,k}$ of Ord_n, plus the following restricted version of the clauses M_i:

$$\bigvee_{j \in N(i)} x_{i,j} \qquad \text{for } 1 \leq i \leq n \qquad (M_i')$$

Here $N(i)$ denotes the neighborhood of i in G, i.e.., the set $\{j \in V; \{i,j\} \in E\}$. The formula requires that for every vertex, there is a larger one in the ordering among its neighbors. Thus in this notation, the formula Ord_n is $\mathrm{Ord}(K_n)$ for the complete graph K_n on n vertices. If the graph G has maximum degree $d \geq 3$, then $\mathrm{Ord}(G)$ is a d-CNF.

A graph G on n vertices is called *ϵ-neighborly*, if for all pairs of disjoint subsets $A, B \subseteq V$ with $|A|, |B| \geq \epsilon n$ there is an edge $\{a, b\} \in E$ with $a \in A$ and $b \in B$. A lower bound on the resolution width of $\mathrm{Ord}(G)$ depending on the neighborliness of G was shown by Segerlind et al. [2004]:

Theorem 11. *If* G *is a connected graph on* n *vertices that is* ϵ*-neighborly for* $0 < \epsilon < 1/3$*, then* $\mathrm{Ord}(G)$ *requires resolution width at least* $(\frac{1-3\epsilon}{6})n$.

The following lemma follows from known results about expander graphs that can e.g. be found in the book of Alon and Spencer [2002, Section 9.2].

Lemma 12. *There is a constant* d *such that for every* n*, there is a graph* G_n *on* n *vertices that has maximum degree* d *and is* $1/6$*-neighborly.*

For these graphs G_n, the formula $\mathrm{Ord}(G_n)$ is a d-CNF that requires resolution width $n/12$. By invoking Theorem 6 with $k = n/36$ we obtain the following lower bound for n large enough that $k \geq d$:

Corollary 13. *For sufficiently large* n*, every* $\mathrm{RTL}(n/36)$*-refutation of* $\mathrm{Ord}(G_n)$ *for the graphs* G_n *is of size at least* $2^{n/36}$.

As above, it follows that a DLL algorithm with clause learning requires exponential time to solve $\mathrm{Ord}(G_n)$ when only clauses of width $n/36$ are learned. On the other hand, short regular resolution refutations of $\mathrm{Ord}(G_n)$ are contained in the refutations of Ord_n. From those, one can again construct a run of a DLL algorithm with arbitrary clause learning on $\mathrm{Ord}(G_n)$ in polynomial time. Hence the formulas $\mathrm{Ord}(G_n)$ are another example of formulas of constant width for which learning wide clauses is necessary to solve them efficiently.

References

[Alekhnovich *et al.*, 2007] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. *Theory of Computing*, 3:81–102, 2007.

[Alon and Spencer, 2002] N. Alon and J. Spencer. *The Probabilistic Method*. John Wiley and Sons, 2002.

[Atserias and Dalmau, 2008] Albert Atserias and Victor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74:323–334, 2008.

[Atserias *et al.*, 2009] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause learning algorithms with many restarts and bounded-width resolution. In Oliver Kullmann, editor, *Theory and Practice of Satisfiability Testing – SAT 2009*, pages 114–127. Springer LNCS 5584, 2009.

[Bayardo Jr. and Schrag, 1997] Roberto J. Bayardo Jr. and Robert C. Schrag. Using CSP look-back techniques to solver real-world SAT instances. In *Proc. 14th Natl. Conference on Artificial Intelligence*, pages 203–208, 1997.

[Beame *et al.*, 2004] Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.

[Ben-Sasson and Wigderson, 2001] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow — resolution made simple. *Journal of the ACM*, 48:149–169, 2001.

[Bonet and Galesi, 2001] Maria Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, 2001.

[Buss *et al.*, 2008] Samuel R. Buss, Jan Hoffmann, and Jan Johannsen. Resolution trees with lemmas: Resolution refinements that characterize DLL algorithms with clause learning. *Logical Methods in Computer Science*, 4(4), 2008.

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

[Gomes *et al.*, 1997] Carla P. Gomes, Bart Selman, and Nuno Crato. Heavy-tailed distributions in combinatorial search. In Gert Smolka, editor, *Principles and Practice of Constraint Programming - CP97*. Springer LNCS 1330, 1997.

[Hertel *et al.*, 2008] Philipp Hertel, Fahiem Bacchus, Toniann Pitassi, and Allen van Gelder. Clause learning can effectively p-simulate general propositional resolution. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the 23rd AAAI Conference on Artificial Intelligence, AAAI 2008*, pages 283–290. AAAI Press, 2008.

[Hoffmann, 2007] Jan Hoffmann. Resolution proofs and DLL-algorithms with clause learning. Diploma Thesis, LMU München, 2007.

[Iwama and Miyazaki, 1999] Kazuo Iwama and Shuichi Miyazaki. Tree-like resolution is superpolynomially slower than dag-like resolution for the pigeonhole principle. In *Proceedings of the 10th International Symposium on Algorithms and Computation (ISAAC)*, pages 133–142, 1999.

[Johannsen, 2009] Jan Johannsen. An exponential lower bound for width-restricted clause learning. In Oliver Kullmann, editor, *Theory and Practice of Satisfiability Testing – SAT 2009*, pages 128–140. Springer LNCS 5584, 2009.

[Kolaitis and Vardi, 1995] Phokion G. Kolaitis and Moshe Y. Vardi. On the expressive power of Datalog: Tools and a case study. *Journal of Computer and System Sciences*, 51(1):110–134, 1995.

[Krishnamurthy, 1985] Balakrishnan Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22:253–274, 1985.

[Marques-Silva and Sakallah, 1996] João P. Marques-Silva and Karem A. Sakallah. GRASP - a new search algorithm for satisfiability. In *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 220–227, 1996.

[Pipatsrisawat and Darwiche, 2009] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers with restarts. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming (CP09)*, pages 654–668, 2009.

[Segerlind *et al.*, 2004] Nathan Segerlind, Samuel R. Buss, and Russell Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM Journal on Computing*, 33(5):1171–1200, 2004.

[Stålmarck, 1996] Gunnar Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33:277–280, 1996.

[Tseitin, 1968] G.S. Tseitin. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125, 1968.

[Zhang *et al.*, 2001] Lintao Zhang, Conor F. Madigan, Matthew W. Moskewicz, and Sharad Malik. Efficient conflict driven learning in a Boolean satisfiability solver. In *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 279–285, 2001.