Lower bounds for width-restricted clause learning on small width formulas

Eli Ben-Sasson[1] and Jan Johannsen[2]

[1] Computer Science Department, Technion – Israel Institute of Technology, Haifa, Israel
[2] Institut für Informatik, LMU München, Munich, Germany

Abstract. It has been observed empirically that clause learning does not significantly improve the performance of a SAT 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 resolution width lower bounds. This yields the first lower bounds on width-restricted clause learning for formulas in 3-CNF.

1 Introduction

Most SAT solvers are based on extensions of the basic backtracking procedure known as the DLL algorithm [10]. The recursive procedure is called for a formula F in conjunctive normal form and a partial assignment α (which is empty in the outermost call). If α satisfies F, then it is returned, and if α causes a conflict, i.e., falsifies a clause in F, then the call fails. Otherwise a variable x unset by α is chosen according to some heuristic, and the procedure is called recursively twice, with α extended by $x := 1$ and by $x := 0$. If one recursive call returns a satisfying assignment, then it is returned, otherwise — if both recursive calls fail — the call fails.

Contemporary SAT solvers employ several refinements and extensions of the basic DLL algorithm. One of the most successful of these extensions is *clause learning* [22], which works as follows: When the procedure encounters a conflict, then a sub-assignment α' of α that suffices to cause this conflict is computed. This sub-assignment α', thought of as the reason for the conflict, 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 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 heuristics for clause learning 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 heuristics are not very helpful, i.e., learning only short 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 [9] for the well-known pigeonhole principle clauses PHP_n. These formulas require time $2^{\Omega(n \log n)}$ 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 [15] for a a set of clauses Ord_n based on the ordering principle on n elements. 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 [14, 8] on the time for solving the respective formulas by DLL algorithms without clause learning.

In these previous lower bounds, the hard example formulas 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 length of refutations in a certain resolution based propositional proof system called RTL (see Section 2). The relationship of this proof system to the DLL algorithm with clause learning has been established in several earlier works [9, 12]. 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 [15] for the formulas Ord_n.

The lower bound for clause learning algorithms on formulas requiring large resolution width is somewhat dual to a result of Atserias et al. [4], 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 informally describe our proof method, see Section 2 for precise definitions of the terms appearing below. Let F be an unsatisfiable formula in conjunctive normal form (CNF) over n variables. For the sake of simplicity assume that F is a 3-CNF, i.e., each clause in F contains at most 3 literals. Suppose furthermore that F requires large resolution refutation width, i.e., every resolution refutation of F must contain a clause with a large number w of literals, where e.g. $w \approx \sqrt{n}$. Finally, suppose we try to solve F using a DLL algorithm augmented with clause learning, where the width of learned clauses is limited to

be less than w, say $w/3$. In other words, the maximal width of a learned clause is significantly smaller than the minimal refutation width of F.

Inspired by the combinatorial characterization of resolution width due to Atserias and Dalmau [3] we classify the learned clauses of small width into two categories. The *useless* clauses are those that can be derived from F via a resolution derivation of width less than w, whereas the *useful* clauses are those that can only be derived by going through a clause of width at least w. Our main observation (Theorem 6) roughly says that learning useless clauses will not significantly reduce the running time needed to obtain the first useful clause. In fact, we show that $2^{w/3}$ steps will be needed before our algorithm learns its first useful clause.

Using known constructions [8, 21, 3] of families of unsatisfiable 3-CNF formulas that have short resolution refutations of polynomial in n size, but which require large refutation width of about $w \approx \sqrt{n}$, we obtain in Section 5 a number of results which show that, in certain cases, width-restricted clause learning DLL algorithms will require exponentially longer running time than clause learning algorithms with unrestricted width.

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 [7] 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

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

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

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

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 $C\restriction\rho$ from $F\restriction\rho$ of size at most s.*

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

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 [24], but regular dag-like refutations can necessarily be exponentially longer than general ones [1].

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 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. [9], 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 [9] 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* [22], *all-UIP*, *decision* [25] and *rel-sat* [5]. A variant of DLL-L-UP which incorporates these learning strategies and also allows for non-chronological backtracking [5] was described by Hoffmann [13] and shown to be likewise simulated by WRTI.

In addition to clause learning, most state-of-the-art satisfiability solvers also use *restarts* [11], therefore their performance is 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 [6] and more recently also for most natural learning strategies [20]. However, these simulations of general dag-like resolution proofs, as well as the clause learning algorithm of Atserias et al. [4] 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.

It follows from the mentioned results of Buss et al. [9] 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 RTL(k). In this work we prove lower bounds on the size of RTL(k)-refutations, which thus yield lower bounds on the runtime of DLL algorithms with width-restricted clause-learning.

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 [3], 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 [3] 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 [16] 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*

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

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 $x \in X \setminus \rho$ 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 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* 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 RTL(k)-refutation of F. By Theorem 3, there is a w-system of restrictions $\mathcal{H}$ that avoids F.

Let C be the first clause in R 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 R_C is an RTL(k)-derivation of C from F. Let G be the set of lemmas used in R_C, so R_C is a tree-like resolution derivation of C from $F \wedge G$, and thus $R' := R_C\restriction\rho$ 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 the lemmas above 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 $R_C\lceil\rho$, 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 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 $i \neq j$:

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

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 [17] as potential hard example formulas for resolution, but short regular resolution refutations for them were constructed by Stålmarck [23].

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 [9] yields WRTI-refutations of Ord_n of polynomial size. On the other hand, a lower bound for RTL(k)-refutations of Ord_n was shown by the second author [15]:

Theorem 8. *For* $k < n/4$, *every* 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 [8] 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. We show a larger lower bound by exhibiting a suitable system of restrictions:

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

For ease of notation, we denote the clauses in the 3-CNF expansion $E_3(M_i)$ of the formula M_i as follows:

$$y_{i,0} \qquad \cdots \qquad \bar{y}_{i,i-1} \vee x_{i,i+1} \vee y_{i,i+1} \qquad \cdots \qquad \bar{y}_{i,n}$$

For a non-empty set $D \subseteq \{1, \ldots, n\}$, a total ordering $\prec$ on D and a partial mapping $s : D \to D$ with the properties

- $s(i)$ is defined for every $i \in D$ except $\max_{\prec} S$,
- $i \prec s(i)$ for every $i \in \operatorname{dom} s$,

we define a restriction $\rho(D, \prec, s)$ as follows:

$$x_{i,j} \;\mapsto\; \begin{cases} 1 & \text{if } i,j \in D \text{ and } i \prec j \\ 0 & \text{if } i,j \in D \text{ and } j \prec i \end{cases}$$

$$y_{i,j} \;\mapsto\; \begin{cases} 1 & \text{if } i \in D,\ s(i) \text{ is defined and } j \geq s(i) \\ 0 & \text{if } i \in D,\ s(i) \text{ is defined and } j < s(i) \end{cases}$$

and is undefined in all other cases. Now let $\mathcal{H}_{\mathrm{ord}}$ be the set of restrictions σ such that $|\sigma| \leq n/2$ and $\sigma \subseteq \rho(D, \prec, s)$ for some subset $D \subseteq \{1, \ldots, n\}$, ordering $\prec$ on D and suitable mapping $s : D \to D$. Theorem 9 now follows immediately from the following lemma by Theorem 3.

Lemma 10. *$\mathcal{H}_{\mathrm{ord}}$ is an $n/2$-system of restrictions that avoids $E_3(\mathrm{Ord}_n)$.*

Proof. Obviously, $\mathcal{H}_{\mathrm{ord}}$ is non-empty, and the size bound $|\sigma| \leq n/2$ for all $\sigma \in \mathcal{H}_{\mathrm{ord}}$ and downward closure hold by definition. The clauses $A_{i,j}$, $T_{i,j}$ and $\Delta_{i,j,k}$ are avoided since the variables $x_{i,j}$ are set according to the ordering $\prec$. The clauses in $E_3(M_i)$ containing a variable $x_{i,j}$ for $j \neq s(i)$ are avoided since both extension variables are set to the same value, and one of them occurs positively and the other negatively. The clause in $E_3(M_i)$ containing $x_{i,s(i)}$ is avoided since this variable cannot be set to 0.

It remains to show that $\mathcal{H}_{\mathrm{ord}}$ has the extension property. If $\sigma \in \sigma \in \mathcal{H}_{\mathrm{ord}}$ is of size $|\sigma| < n/2$, then there is a set D of size $|D| \leq n-2$, an ordering $\prec$ on D

and a mapping $s : D \to D$ such that $\sigma \subseteq \rho := \rho(D, \prec, s)$. Let $v \notin \operatorname{dom}\sigma$ be a variable left unset by σ.

If $v = x_{i,j}$, then we set $D' := D \cup \{i, j\}$. If $D' = D$, then $\rho(v)$ is already defined, and we set $\prec'=\prec$ and $s' = s$. Otherwise, if $i \in D' \setminus D$, we extend $\prec$ and s to $\prec'$ and s' by setting $i \prec' k$ for every $k \in D$ and $s'(i) := \min_{\prec} D$, and similarly for j.

If $v = y_{i,j}$ and $i \in D$, then $\rho(v)$ is already defined unless $i = \max_{\prec} D$. In the latter case, we pick an arbitrary $k \notin D$ and set $D' := D \cup \{k\}$, extend $\prec$ to $\prec'$ by setting $i \prec' k$ and s to s' by setting $s'(i) = k$.

If $v = y_{i,j}$ and $i \notin D$, then we set $D' := D \cup \{i\}$, and we extend $\prec$ to $\prec'$ by setting $i \prec' k$ for all $k \in D$ and s to s' by setting $s'(i) = \min_{\prec} D$.

In all cases $\rho' := \rho(D', \prec', s')$ is an extension $\rho' \supseteq \rho$ with $v \in \operatorname{dom}\rho'$. Let $\sigma' := \sigma \cup \{(v, \rho'(v))\}$, then we have $|\sigma'| \leq n/2$, and $\sigma' \subseteq \rho'$, hence we have found $\sigma' \in \mathcal{H}_{\mathrm{ord}}$ with $v \in \operatorname{dom}\sigma'$. □

By Theorem 6, a lower bound for 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 11. *For $n \geq 18$, every* 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 RTL(k)-refutation of Ord_n of size s can be converted into an RTL(k)-refutation of $E_3(\mathrm{Ord}_n)$ of size sn. Hence the Corollary 11 also yields an easier proof of a slightly weaker variant of the lower bound from Theorem 8: every 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. [21]. 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. [21]:

Theorem 12. *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 [2, Section 9.2]. In what follows a family of graphs $\{G_n; n \in \mathbb{N}\}$ is said to be *explicitly constructible* if there exists a polynomial time Turing machine that on input 1^n outputs a description (say, by means of its adjacency matrix) of the graph G_n.

Lemma 13. *For every $0 < \epsilon < 1/3$ there is a constant $d = O(1/\epsilon^2)$ such that there is an explicitly constructible family of ϵ-neighborly graphs $\{G_n; n \in \mathbb{N}\}$ on n vertices of maximal degree d.*

Proof. As explained in the mentioned book [2, Section 9.2] (and using the notation there), the works of Lubotzky et al. [18] and of Margulis [19] explicitly construct for every integer d of the form $d = p+1$, where p is a prime congruent to 1 modulo 4, and for every sufficiently large n, a d-regular expander graph G_n, with all eigenvalues of the adjacency matrix except for the largest being bounded in absolute value by $2\sqrt{d-1}$ (such graphs are known as "Ramanujan" expander graphs). For such graphs one may apply Corollary 9.2.5 in the book [2] which implies that every two disjoint subsets of the vertices of G_n of size at least $\frac{2n}{\sqrt{d}}$ must be connected by an edge, i.e., G_n is $\frac{2}{\sqrt{d}}$-neighborly. □

The lemma yields, e.g., a family of graphs G_n on n vertices of maximal degree $d = 150$ that are 1/6-neighborly, and 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 $k \geq 150$:

Corollary 14. *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 small width for which learning wide clauses is necessary to solve them efficiently.

Dense Linear Ordering Principle

The dense linear ordering principle yields another family of formulas that have short regular resolution refutations, but require large resolution width. It says that a finite linear ordering cannot be dense. It gives rise to an unsatisfiable set of clauses DLO_n, in the variables $x_{i,j}$ representing the ordering as in Ord_n, and additional variables $z_{i,j,k}$ intended to express that j is between i and k in the ordering. It consists of the clauses $A_{i,j}$, $T_{i,j}$ and $\Delta_{i,j,k}$ of Ord_n, plus new clauses containing the variables $z_{i,j,k}$:

$$\begin{aligned}
&\bar{x}_{i,j} \vee \bar{x}_{j,k} \vee z_{i,j,k} && \text{for } 1 \le i,j,k \le n \text{ pairwise distinct} \\
&\bar{z}_{i,j,k} \vee x_{i,j} && \text{for } 1 \le i,j,k \le n \text{ pairwise distinct} \\
&\bar{z}_{i,j,k} \vee x_{j,k} && \text{for } 1 \le i,j,k \le n \text{ pairwise distinct} \\
&\bar{x}_{i,k} \vee \bigvee_{1 \le j \le n, j \ne i,k} z_{i,j,k} && \text{for } 1 \le i,k \le n \text{ with } i \ne k \qquad (D_{i,k})
\end{aligned}$$

The first three groups of clauses enforce that the values of the variables $z_{i,j,k}$ define the relation "*j is between i and k*", and the clause $D_{i,k}$ states that if $i \prec k$, then there exists an element between i and k. Therefore the formula DLO_n expresses that there is a dense linear ordering on n points, and is thus unsatisfiable.

Atserias and Dalmau [3] show a lower bound on the resolution width of the 3-CNF expansion $E_3(\mathrm{DLO}_n)$ of the dense linear ordering principle:

Theorem 15. *The formula $E_3(\mathrm{DLO}_n)$ requires resolution width at least $n/3$.*

Using Theorem 6 with $k = n/9$, it follows:

Corollary 16. *For $n \ge 27$, every $\mathrm{RTL}(n/9)$-refutation of $E_3(\mathrm{DLO}_n)$ is of size at least $2^{n/9}$.*

Again, it follows that a DLL algorithm with clause learning requires exponential time to solve $E_3(\mathrm{DLO}_n)$ when only clauses of width $n/9$ are learned. On the other hand, short resolution refutations of DLO_n and of $E_3(\mathrm{DLO}_n)$ are given by Atserias and Dalmau [3], and these refutations are easily seen to be regular. Hence there is a run of a DLL algorithm with arbitrary clause learning on $E_3(\mathrm{DLO}_n)$ in polynomial time, and thus learning wide clauses is necessary to solve these formulas efficiently.

References

1. M. Alekhnovich, J. Johannsen, T. Pitassi, and A. Urquhart. An exponential separation between regular and general resolution. *Theory of Computing*, 3:81–102,

2007. Preliminary Version in Proc. 34th ACM Symposium on Theory of Computing, 2002.
2. N. Alon and J. Spencer. *The Probabilistic Method*. John Wiley and Sons, 2002.
3. A. Atserias and V. Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74:323–334, 2008. Preliminary version in Proc. 18th IEEE Conference on Computational Complexity, 2003.
4. A. Atserias, J. K. Fichte, and M. Thurley. Clause learning algorithms with many restarts and bounded-width resolution. In O. Kullmann, editor, *Theory and Practice of Satisfiability Testing – SAT 2009*, pages 114–127. Springer LNCS 5584, 2009.
5. R. J. Bayardo Jr. and R. 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.
6. P. Beame, H. A. Kautz, and A. Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.
7. E. Ben-Sasson and A. Wigderson. Short proofs are narrow — resolution made simple. *Journal of the ACM*, 48, 2001. Preliminary Version in Proc. 31st ACM Symposium on Theory of Computing, 1999.
8. M. L. Bonet and N. Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, 2001.
9. S. R. Buss, J. Hoffmann, and J. Johannsen. Resolution trees with lemmas: Resolution refinements that characterize DLL algorithms with clause learning. *Logical Methods in Computer Science*, 4(4), 2008.
10. M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.
11. C. P. Gomes, B. Selman, and N. Crato. Heavy-tailed distributions in combinatorial search. In G. Smolka, editor, *Principles and Practice of Constraint Programming - CP97*. Springer LNCS 1330, 1997.
12. P. Hertel, F. Bacchus, T. Pitassi, and A. van Gelder. Clause learning can effectively p-simulate general propositional resolution. In D. Fox and C. P. Gomes, editors, *Proceedings of the 23rd AAAI Conference on Artificial Intelligence, AAAI 2008*, pages 283–290. AAAI Press, 2008.
13. J. Hoffmann. Resolution proofs and DLL-algorithms with clause learning. Diploma Thesis, LMU München, 2007.
14. K. Iwama and S. 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.
15. J. Johannsen. An exponential lower bound for width-restricted clause learning. In O. Kullmann, editor, *Theory and Practice of Satisfiability Testing – SAT 2009*, pages 128–140. Springer LNCS 5584, 2009.
16. P. G. Kolaitis and M. Y. Vardi. On the expressive power of Datalog: Tools and a case study. *Journal of Computer and System Sciences*, 51(1):110–134, 1995. Preliminary version in Proc. 9th ACM Symposium on Principles of Database Systems, 1990.
17. B. Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22:253–274, 1985.
18. A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

19. G. A. Margulis. Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Problems of Information Transmission*, 24:39–46, 1988.
20. K. Pipatsrisawat and A. 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.
21. N. Segerlind, S. R. Buss, and R. Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM Journal on Computing*, 33(5):1171–1200, 2004. Preliminary version in Proc. 43rd IEEE Symposium on Foundations of Computer Science, 2002.
22. J. P. M. Silva and K. A. Sakallah. GRASP - a new search algorithm for satisfiability. In *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 220–227, 1996.
23. G. Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33:277–280, 1996.
24. G. Tseitin. On the complexity of derivation in propositional calculus. *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125, 1968.
25. L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. 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.