An exponential lower bound for width-restricted clause learning

Jan Johannsen
Institut für Informatik
LMU München

April 15, 2009

Abstract

It has been observed empirically that clause learning does not significantly improve the performance of a satisfiability solver when restricted to learning short clauses only. This experience is supported by a lower bound theorem: an unsatisfiable set of clauses, claiming the existence of an ordering of n points without a maximum element, can be solved in polynomial time when learning arbitrary clauses, but it is shown to require exponential time when learning only clauses of size at most $n/4$. The lower bound is of the same order of magnitude as a known lower bound for backtracking algorithms without any clause learning. It is shown by proving lower bounds on the proof length in a certain resolution proof system related to clause learning.

1 Introduction

Most contemporary SAT solvers are based on extensions of the basic backtracking procedure known as the DLL-algorithm [6]. One of the most successful of these extensions is clause learning [11], which works roughly as follows: When the backtracking algorithm encounters a conflict, i.e., a clause falsified by the current partial assignment α, then a sub-assignment α' of α that suffices to cause this conflict is computed. This sub-assignment α', 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 the algorithm can later backtrack earlier when again a partial assignment extending α' occurs in another branch of the search tree, since then the added clause C becomes falsified and thus causes a conflict.

When clause learning is implemented, a heuristic is needed to decide which learnable clauses to actually keep in memory, as learning a large

number of clauses leads to excessive memory usage, which slows the algorithm down rather than helping it. An obvious simple heuristic is to learn only short clauses, i.e., to set a threshold (possibly depending on the input clauses), and to keep in memory only clauses whose size does not exceed the threshold.

Researchers who have experimented with heuristics for clause learning, e.g. the author himself or Letz [9], have experienced that this simple heuristic is not very helpful, i.e., learning only short clauses does not significantly improve the performance of a DLL algorithm for hard formulas The present work aims at supporting this experience with a rigorous mathematical analysis in the form of a lower bound theorem.

In earlier work [5], we have shown such a lower bound 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 the weakness of the heuristic, it is not fully satisfactory, since even with arbitrary learning, the time required is exponential in n, it just takes still more time – about $n!$ – to solve when learning short clauses only.

Here we provide another example using a set of clauses Ord_n based on the ordering principle. 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. This lower bound is asymptotically the same as the known exponential lower bound [4] on the time for solving Ord_n by DLL algorithms without clause learning.

The lower bounds on the run-time are shown by proving the same lower bounds on the length of refutations in a certain propositional proof system. The relationship of this proof system to the DLL algorithm with clause learning has been established in several earlier works [5, 7].

2 Preliminaries

A *literal* is a variable x or a negated variable $\bar{x}$, the former are *positive* literals and the latter *negative* literals. A *clause* is a disjunction $C = a_1 \vee \ldots \vee a_k$ of literals a_i, its *width* is $w(C) = k$, the number of literals in it. We identify a clause with the set of literals occurring in it, even though for clarity we still write it as a disjunction. A clause is *negative* if it contains no positive literals. 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 $\{C_1, \ldots, C_m\}$.

We consider refutation systems for formulas in CNF based on the resolution rule, which are well-known to be strongly related to DLL algorithms. The proof systems under consideration 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, and with every node v labeled with a clause C_v such that:

1. The sink is labeled with C.
2. If a node v has one predecessor v', then C_v follows from $C_{v'}$ by the weakening rule.
3. If a node v has two predecessors v_1, v_2, then C_v follows from C_{v_1} and C_{v_2} by the resolution rule.
4. A source node v is labeled by a clause C in F.

The *size* of a resolution derivation is the number of nodes in the dag. A *resolution refutation* of F is a resolution derivation of the empty clause from F. We call a derivation *tree-like* if the underlying unlabeled dag is a tree, otherwise we may call it *dag-like* for emphasis.

Note that the weakening rule is redundant in tree-like and dag-like resolution refutations: its uses can be eliminated from a refutation without increasing the size. This may not be the case for the proof system we define below.

A resolution derivation is called *regular* if on every path through the dag, each variable is eliminated at most once. Regularity is not an essential restriction on tree-like resolution since minimal tree-like refutations are always regular [13], but regular dag-like refutations can necessarily be exponentially longer than general ones [1].

Regular tree-like resolution exactly corresponds to the DLL algorithm by the following well-known correspondence: the run of a DLL-algorithm on an unsatisfiable formula F forms a regular, tree-like resolution refutation of F without use of the weakening rule. Since the weakening rule is redundant in tree-like resolution proofs, the converse direction holds as well.

The proof system studied in this work are *resolution trees with lemmas* (RTL), which are defined as follows: An RTL-derivation of C from F is defined like a tree-like resolution derivation of C from F, but here a node with in-degree 2 has a distinguished *left* and *right* predecessor. Then the clause 4 of the definition liberalized to:

4a. A source node ν is labeled by a clause D in F, or by a clause C labelling some node $\nu' \prec \nu$. In the latter case we call C a *lemma*.

Here $\prec$ denotes the post-ordering of the tree, i.e., the order in which the nodes of the tree are visited by a post-order traversal, which at a node ν with two predecessors first recursively traverses the left subtree, i.e., the subtree rooted at the left predecessor of ν, then recursively traverses the right subtree, and then visits ν itself.

An RTL-derivation is an RTL(k)-derivation if every lemma C is of width $w(C) \leq k$. An RTL-derivation of the empty clause from F is an RTL-refutation of F. Note that RTL is equivalent to dag-like resolution and RTL(0) is equivalent to tree-like resolution.

A subsystem WRTI of RTL has been described by Buss et al. [5] which corresponds to a general formulation of the DLL algorithm with clause learning. This system WRTI imposes the regularity restriction on derivations, and does not include the full weakening rule, but incorporates some amount of weakening into a generalized resolution inference rule, the so-called w-resolution rule. It also restricts further the structure of sub-derivations of clauses that can be used as lemmas, which have to be derived by input resolution derivations. W.r.t. the length of proofs, WRTI lies between regular and general dag-like resolution.

The size of a refutation of an unsatisfiable formula F in WRTI has been shown [5] to be polynomially related to the runtime of a schematic algorithm DLL-L-UP on F. This schema DLL-L-UP subsumes all commonly used clause learning strategies, including *first-UIP* [11], *all-UIP*, *decision* [15] and *rel-sat* [2], but is slightly more general than a DLL algorithm with clause learning by being non-greedy in the sense that it can continue branching even after a conflict was reached. In the simulation of clause learning by WRTI, the clauses learned by the algorithm are those clauses used as lemmas in the refutation.

A different system with similar properties was described by Hertel et al. [7], building on earlier work of van Gelder [14], which can likewise be seen as a subsystem of RTL.

It follows 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

k, then the refutation is in $\mathrm{RTL}(k)$. In the following we prove lower bounds on the size of refutations in $\mathrm{RTL}(k)$, which thus readily translate into lower bounds on the runtime of DLL with width-restricted clause-learning.

A common tool in proof complexity is to consider formulas under a partial assignment, called *restriction* in this context. We shall need a slightly more general notion of restriction in this work.

Let X be a set of variables. A *restriction with renaming* is a (total) function $\rho : X \to X \cup \{0, 1\}$. The function ρ is extended to literals by setting

$$\rho(\bar{x}) := \begin{cases} 1 & \text{if } \rho(x) = 0 \\ 0 & \text{if } \rho(x) = 1 \\ \overline{\rho(x)} & \text{if } \rho(x) \in X\,. \end{cases}$$

For a clause C in variables X, we define

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

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

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

where the empty conjunction is identified with 1.

Just like ordinary restrictions, the more general renaming restrictions preserve proofs in most propositional proof systems. We state this fact here only for resolution.

Proposition 1. *Let R be a (tree-like) resolution proof of C from F of size s, and ρ a restriction with renaming. Then there is a (tree-like) resolution proof R' of $C\restriction\rho$ from $F\restriction\rho$ of size at most $2s$.*

The proposition is shown by a straightforward induction along the proof R, the proof will not be given here, as we will prove a special case that we actually use below.

In the following we just use the word restriction for restrictions with renaming, since ordinary restrictions do not occur in this work.

3 The ordering principle

The ordering principle expresses the fact that every finite total 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 $1 \leq i < j, k \leq n$ with $j \neq k$	$(\Delta_{i,j,k})$
$\bigvee_{j \in [n] \setminus \{i\}} x_{i,j}$	for $1 \leq i \leq n$	(M_i)

Let R be the relation on $[n]$ given by an assignment to the variables, so that $i \mathrel{R} j$ holds iff $x_{i,j}$ is set to 1. The clauses $A_{i,j}$ and $T_{i,j}$ state that for every i and j, either $i \mathrel{R} j$ or $j \mathrel{R} i$ holds, but not both. The clause $\Delta_{i,j,k}$ state that there are no cycles of length 3 in R, which modulo the first two families of clauses is equivalent to R being transitive. Thus the first three clause sets state that R is a total ordering. The clauses M_i then state that this ordering has no maximal element, therefore the formula is unsatisfiable.

The formulas Ord_n were introduced by Krishnamurthy [8] as potential hard example formulas for resolution, but short regular resolution refutations for them were constructed by Stålmarck [12].

Proposition 2. *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 [5] yields WRTI-refutations of Ord_n of polynomial size. From these, it is straightforward to construct a polynomial length run of a DLL algorithm with clause learning on Ord_n, making the branching and learning decisions suggested by the refutation.

On the other hand, the following lower bound for tree-like resolution refutations of Ord_n was shown by Bonet and Galesi [4]. It implies that a DLL algorithm without clause learning requires exponential time to solve these formulas.

Theorem 3. *Every tree-like resolution refutation of* Ord_n *is of size* $2^{\Omega(n)}$.

More precisely, the lower bound proved by Bonet and Galesi is $\Omega(2^{n/6})$. We shall prove a larger lower bound of $\Omega(2^{n/2})$ below. Our main result is a lower bound on the size of $\mathrm{RTL}(k)$-refutations of the formulas Ord_n.

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

It follows that a DLL algorithm with learning requires exponential time to solve these formulas, when learning is restricted to clauses of width less than $n/4$.

The idea of the proof is similar to that of the mentioned lower bound for the pigeonhole principle PHP_n [5]: the goal is to show that a long derivation is required to obtain a clause that is short enough to be used as a lemma. To prove this, look at the first sufficiently short clause C, and find a restriction ρ falsifying C. Then the derivation of C, restricted by ρ, is a tree-like resolution refutation of $\mathrm{PHP}_{n'}$ for some $n' < n$, and therefore needs to be large by a known lower bound.

This strategy does not quite work here directly, since from Ord_n short clauses can be derived very quickly. Therefore we single out a class of useful clauses, and show that any refutation can be transformed so that only these useful clauses are used as lemmas, in Section 5.

After that, we again look at the first clause used as a lemma, and find a restriction falsifying it. Thereby we obtain a tree-like refutation of a smaller instance of the ordering principle, which needs to be large by a known lower bound. A class of restrictions that makes this construction possible is defined below.

The argument becomes simpler if the proof is first brought into a normal form that contains only negative clauses; this is done in Section 4. Finally, in Section 6, everything is put together to prove the theorem.

As mentioned, we need to define a class of restrictions that preserve the ordering principle clauses, similar to the matching restrictions that preserve the pigeonhole principle formulas, but in contrast to those we require restrictions with renaming. For a non-empty set $S \subseteq [n]$ and a total ordering $\prec$ on S, we define the *ordering restriction* $\rho(S, \prec)$ by

$$\rho(S,\prec): \quad x_{i,j} \mapsto \begin{cases} 1 & \text{if } i,j \in S \text{ and } i \prec j \\ 0 & \text{if } i,j \in S \text{ and } j \prec i \\ x_{s,j} & \text{if } i \in S \text{ and } j \notin S \\ x_{i,s} & \text{if } i \notin S \text{ and } j \in S \\ x_{i,j} & \text{otherwise,} \end{cases}$$

where $s \in S$ is arbitrary but fixed, e.g. $s := \max S$. We let σ range over ordering restrictions, and for $\sigma = \rho(S, \prec)$ we let $|\sigma| := |S|$.

The main property of ordering restrictions is that they preserve the ordering principle formulas.

Proposition 5. *For every ordering restriction σ with $|\sigma| \geq 1$,*

$$\mathrm{Ord}_n\restriction\sigma \;=\; \mathrm{Ord}_{n-|\sigma|+1}\,.$$

Proof. We shall see that the restriction of every clause from Ord_n by $\sigma = \rho(S, \prec)$ with $|S| \geq 1$ is again one of the clauses from Ord_n, with indices from $[n] \setminus S \cup \{s\}$. Thus after a renaming of variables we obtain the clauses $\mathrm{Ord}_{n-|S|+1}$.

The clauses $T_{i,j}$, $A_{i,j}$ and $\Delta_{i,j,k}$ for $i, j, k \notin S$ remain unaffected by the restriction.

The restriction by σ of the clauses $T_{i,j}$, where $i \in S$ and $j \notin S$ are the clauses $T_{s,j}$, and similarly for $j \in S$ and $i \notin S$. The clauses $T_{i,j}\restriction\sigma$ with $\{i, j\} \subseteq S$ are satisfied. The analogous statements hold for the clauses $A_{i,j}$.

The clauses $\Delta_{i,j,k}\restriction\sigma$ with $i \in S$ and $j, k \notin S$ are $\Delta_{s,j,k}$, and similarly for the other situations where $|\{i, j, k\} \cap S| = 1$.

The clauses $\Delta_{i,j,k}$ with $i, j \in S$ and $k \notin S$ with $j \prec i$ are satisfied by σ, and similarly for the symmetric situations as well as for $\{i, j, k\} \subseteq S$. For $i, j \in S$ with $i \prec j$, the restriction of $\Delta_{i,j,k}$ by σ is $A_{s,k}$, and similarly for the symmetric cases.

Finally, the restriction of M_i for $i \notin S$ is M_i over the smaller domain, for the maximal element i of S under $\prec$ it is M_s, and for other values $i \in S$ it is satisfied. □

4 Negative calculus

We now define a normal form for RTL-derivations from Ord_n, in form of a *negative calculus* NTL that uses only negative clauses.

For a clause C in the variables of Ord_n, we define a negative clause C^N that is equivalent to C w.r.t. ordering restrictions as follows:

$$\begin{aligned} \bar{x}^N_{i,j} &:= \bar{x}_{i,j} \\ x^N_{i,j} &:= \bar{x}_{j,i} \\ C^N &:= \bigvee_{a \in C} a^N \end{aligned}$$

Observe that $w(C^N) \leq w(C)$ for every clause C, but the translated clause can be strictly smaller, e.g., $(x_{1,2} \vee x_{1,3} \vee \bar{x}_{2,1})^N$ is $\bar{x}_{2,1} \vee \bar{x}_{3,1}$. The negative translation Ord^N_n of the ordering principle is the conjunction of the clauses:

$A_{i,j}$	for $1 \leq i < j \leq n$,
$\Delta_{i,j,k}$	for $1 \leq i < j, k \leq n$ with $j \neq k$, and
M^N_i	for $1 \leq i \leq n$.

It is easily seen that the negative translation commutes with ordering restrictions, i.e., for every clause C and ordering restriction σ we have $C^N\restriction\sigma = (C\restriction\sigma)^N$. It follows from Lemma 5 and this fact that ordering restrictions preserve the negative-translated ordering principle:

Corollary 6. *For every ordering restriction σ with $|\sigma| \geq 1$,*

$$\mathrm{Ord}_n^N\restriction\sigma \;=\; \mathrm{Ord}^N_{n-|\sigma|+1}\,.$$

In the negative calculus NTL, the essential positive clauses $T_{i,j}$ in the ordering principle are coded in an inference rule, the *negative inference*:

$$\frac{C \vee \bar{x}_{i,j} \qquad D \vee \bar{x}_{j,i}}{C \vee D}$$

An NTL-derivation is defined exactly as an RTL-derivation, only with the negative inference replacing the resolution inference. An NTL-derivation that does not use any lemmas is called a tree-like negative derivation. Also, an NTL-derivation is an NTL(k)-derivation if every lemma used is of width at most k.

We now provide a translation of RTL-derivations from the ordering principle clauses into the negative calculus that preserves the proof size and the width of lemmas used.

Lemma 7. *If C has an RTL(k)-derivation from Ord_n of size s, then C^N has an NTL(k)-derivation from Ord_n^N of size at most 2s.*

Proof. Let R be an RTL(k)-derivation of C from from Ord_n. We construct an NTL(k)-derivation of C^N of the appropriate size.

For each clause C in Ord_n, the translation C^N is in Ord_n^N, so the claim holds for the axiom leaves. For the lemma leaves, we shall take care in the construction that the clauses C^N for C occurring in R, occur in R^N in the same order, so the lemmas can be used as needed. Also note that since $w(C^N) \leq w(C)$, the lemmas used do not exceed the width bound.

If D is derived by a weakening inference from $C \subseteq D$, and C has a derivation of size $s-1$, then by induction C^N has an NTL(k)-derivation of size at most $2s-2$, and a weakening inference yields $D^N \supseteq C^N$. The size of the obtained derivation is at most $2s-1$, and the ordering of clauses in the derivation is preserved.

Now let $C \vee D$ be derived by a resolution inference from $C \vee x_{i,j}$ and $D \vee \bar{x}_{i,j}$, which are derived by RTL(k)-derivations of size s_1 and s_2, resp., where $s = s_1 + s_2 + 1$. By induction, there are NTL(k)-derivations of $\tilde{C} \vee \bar{x}_{j,i}$ of size at most $2s_1$, and of $\tilde{D} \vee \bar{x}_{i,j}$ of size at most $2s_2$, where $\tilde{C} \subseteq C^N$ and $\tilde{D} \subseteq D^N$. A negative inference then yields $\tilde{C} \vee \tilde{D}$, and by a weakening

inference we obtain $C^N \vee D^N$. Note that C^N might contain $\bar{x}_{j,i}$, or similarly for D^N, thus we can not necessarily obtain $C^N \vee D^N$ immediately by a negative inference. The size of the derivation is at most $2s_1 + 2s_2 + 2 = 2s$, and the ordering is preserved. □

The converse direction also holds, we state it for completeness without proof since we shall not need it here:

Proposition 8. *If* C *has an* NTL(k)*-derivation from* Ord_n^N *of size* s*, then* C *also has an* RTL(k)*-derivation from* Ord_n *of size at most* $6ns$*.*

Negative tree-like derivations are preserved under ordering restrictions. Note that this does not hold for arbitrary restrictions.

Proposition 9. *Let* R *be a tree-like negative derivation of* C *from* F *of size* s*, and* σ *an ordering restriction. There is a tree-like negative derivation* R' *of some subclause* $C' \subseteq C\lceil\sigma$ *from* $F\lceil\sigma$ *of size at most* s*.*

Proof. The proof is by induction of s. If $s = 1$, then R is just the single clause $C \in F$, and hence $C\lceil\sigma$ is in $F\lceil\sigma$, having a derivation of size 1 as well.

If C is derived by weakening from $D \subseteq C$, where D has a derivation of size $s-1$, then by the induction hypothesis there is $D' \subseteq D\lceil\sigma$ having a derivation of size at most $s-1$, from which we obtain $C\lceil\sigma \supseteq D\lceil\sigma \supseteq D'$ by a weakening again.

Now let C be derived from $D_1 = D_1' \vee \bar{x}_{i,j}$ and $D_2 = D_2' \vee \bar{x}_{j,i}$ by a negative inference, with D_i having a derivation of size s_i for $i = 1, 2$ where $s = s_1 + s_2 + 1$. By the induction hypothesis, we have for $i = 1, 2$ a derivation of $D_i'' \subseteq D_i\lceil\sigma$ of size at most s_i. We distinguish three cases.

If $\bar{x}_{i,j}$ does not occur in D_1'', then we obtain $C\lceil\sigma \supseteq D_1'\lceil\sigma \supseteq D_1''$ by weakening, and the resulting derivation is of size at most $s_1 + 1$. The case where $\bar{x}_{j,i}$ does not occur in D_2'' is dual.

Otherwise, we have $D_1'' = \tilde{D}_1 \vee \bar{x}_{i,j}$ and $D_2'' = \tilde{D}_2 \vee \bar{x}_{j,i}$, and we obtain $C' = \tilde{D}_1 \vee \tilde{D}_2 \subseteq D_1'\lceil\sigma \vee D_2'\lceil\sigma = C\lceil\sigma$ by a negative inference, giving a derivation of size at most $s_1 + s_2 + 1 = s$ again. □

In particular, if R is a refutation of F, then R' is a refutation of $F\lceil\sigma$. As usual, we denote R' by $R\lceil\sigma$.

We now prove a lower bound on the size of tree-like negative refutations of the (negative-translated) ordering principle that is slightly larger than the bound obtained from the translation of Theorem 3. Via Lemma 7, it yields the same larger lower bound for tree-like resolution refutations of Ord_n. The proof given here is implicit in the proof of a lower bound for regular resolution refutations of a modification of Ord_n [1].

Lemma 10. *Every tree-like negative refutation of* Ord_n^N *is of size at least* $2^{(n-1)/2}$.

Proof. Let R be a tree-like negative refutation of Ord_n^N. We will define a subtree T of R, and for each node ν in T labeled with the clause C_ν an ordering restriction $\sigma_\nu = \rho(S_\nu, \prec_\nu)$ such that $C_\nu \lceil \sigma_\nu = 0$.

The root of T is the root r of R, and we define $S_r = \emptyset$ and $\prec_r$ as the empty ordering. Since $C_r = 0$, the claim holds.

Now suppose we have defined T up to a node ν with $|\sigma_\nu| \leq n-2$. Since no ordering restriction of size less than n falsifies a clause in Ord_n^N, ν must be an inner node in R.

If ν has a single successor ν', and C_ν is derived by weakening from $C_{\nu'} \subset C_\nu$, then $C_{\nu'} \lceil \sigma_\nu = 0$, so we add ν' to T and set $\sigma_{\nu'} = \sigma_\nu$.

If ν has two successors ν_1 and ν_2, and C_ν is derived by a negative inference

$$\frac{C_{\nu_1} = C \vee \bar{x}_{i,j} \qquad C_{\nu_2} = D \vee \bar{x}_{j,i}}{C_\nu = C \vee D}$$

then we distinguish two cases.

If $i \in S_\nu$ and $j \in S_\nu$, then we add one of the children of ν to T. If $i \prec_\nu j$, then we set $\nu' = \nu_1$, otherwise we set $\nu' = \nu_2$, and we add ν' to T. In either case, by construction we have $C_{\nu'} \lceil \sigma_\nu = 0$, and thus we set $\sigma_{\nu'} = \sigma_\nu$.

If $i \notin S$ or $j \notin S$, then we add both ν_1 and ν_2 to T, and in this case we call ν a *branching node*. We set $S_{\nu_1} = S_{\nu_2} = S_\nu \cup \{i, j\}$. We then choose some extension $\prec_{\nu_1} \supseteq \prec_\nu$ with $i \prec_{\nu_1} j$, and another extension $\prec_{\nu_2} \supseteq \prec_\nu$ with $j \prec_{\nu_2} i$. By construction, we have $C_{\nu_i} \lceil \sigma_{\nu_i} = 0$ and $|S_{\nu_i}| \leq |S_\nu| + 2$ for $i = 1, 2$.

Now every branch in T contains at least $(n-1)/2$ branching nodes, and therefore T and hence R is of size at least $2^{(n-1)/2}$. □

5 Cyclic clauses

For a negative clause C over the variables of Ord_n, let $G(C)$ be the directed graph with vertex set $[n]$ and edges $\{(i, j)\,;\, \bar{x}_{i,j} \in C\}$. A negative clause is *cyclic*, if $G(C)$ contains a (directed) cycle, and acyclic otherwise. It is easily seen that cyclic clauses have short tree-like negative refutations.

Lemma 11. *Any cyclic clause over the variables of* Ord_n *of width* k *has a tree-like negative refutation of size at most* $2\min(k, n)$.

Proof. If $G(C)$ is cyclic, it contains a cycle $i_1, i_2, \ldots i_\ell, i_1$ with $\ell \leq \min(k, n)$. We first show that for every such cycle, the clause

$$\bar{x}_{i_1,i_2} \vee \ldots \vee \bar{x}_{i_{\ell-1},i_\ell} \vee \bar{x}_{i_\ell,i_1}$$

has a negative derivation of length at most $2\ell - 1$. From this clause, the clause C is derived by one weakening inference, hence it has a derivation of length $2\ell \leq 2\min(k, n)$.

We prove the claim by induction on ℓ. For $\ell \leq 3$, this clause is either A_{i_1,i_2} or Δ_{i_q,i_2,i_3}, and hence already in Ord_n^N. Assume the claim holds for ℓ, then by a negative inference we obtain:

$$\frac{\bar{x}_{i_1,i_2} \vee \ldots \vee \bar{x}_{i_{\ell-1},i_\ell} \vee \bar{x}_{i_\ell,i_1} \qquad \bar{x}_{i_1,i_\ell} \vee \bar{x}_{i_\ell,i_{\ell+1}} \vee \bar{x}_{i_{\ell+1},i_1}}{\bar{x}_{i_1,i_2} \vee \ldots \vee \bar{x}_{i_\ell,i_{\ell+1}} \vee \bar{x}_{i_{\ell+1},i_1}}$$

and the length of the resulting derivation is $2\ell - 1 + 2 = 2(\ell + 1) - 1$, which shows the claim. □

It follows that cyclic clauses are useless as lemmas for refuting Ord_n^N.

Lemma 12. *Let R be an $\mathrm{NTL}(k)$-refutation of Ord_n^N of size s. Then there is an $\mathrm{NTL}(k)$-refutation R' of Ord_n^N such that every lemma used in R' is acyclic, and $|R'| \leq 2n \cdot s$.*

Proof. Replace each cyclic lemma used by its derivation of size at most $2n$, which exists by Lemma 11. □

The final ingredient for our proof is the following lemma showing that a short acyclic clause can always be falsified by a small ordering restriction.

Lemma 13. *If C is an acyclic negative clause of width $w(C) \leq k$, then there is an ordering restriction σ of size $|\sigma| \leq 2k$ such that $C\restriction\sigma = 0$.*

Proof. Let S be the set of those $i \leq n$ that are mentioned in C, i.e., such that $\bar{x}_{i,j}$ or $\bar{x}_{j,i}$ occurs in C for some j. Clearly $|S| \leq 2k$. Consider the subgraph G of $G(C)$ induced by S, which only differs from $G(C)$ by omitting isolated vertices. Since C is acyclic, so is G. Let $\prec$ be any topological ordering of G, i.e., a total ordering of S such that $u \prec v$ for every edge (u, v) in G. Then for $\sigma := \rho(S, \prec)$ we have $C\restriction\sigma = 0$ by construction, and $|\sigma| \leq 2k$ as required. □

6 Proof of the lower bound

We are now ready to plug all ingredients together to prove our lower bound result, Theorem 4.

Proof. Let $k < n/4$, and let R be an $\mathrm{RTL}(k)$-refutation of Ord_n of size s. By Lemma 7, there is an $\mathrm{NTL}(k)$-refutation R^N of Ord_n^N of size $|R^N| \leq 2s$. Lemma 12 then yields an $\mathrm{NTL}(k)$-refutation R' of Ord_n^N with only acyclic lemmas, of size $|R'| \leq 4ns$.

Let C be the first clause in R' that is used as a lemma. Then the subtree R'_C of R' rooted at C is a tree-like negative derivation of C from Ord_n^N, of size $|R'_C| \leq 4ns$. Since C is acyclic, from Lemma 13 we obtain an ordering restriction σ of size $|\sigma| \leq 2k < n/2$ such that $C\lceil\sigma = 0$, and Proposition 9 yields a tree-like negative refutation $\tilde{R} := R'_C\lceil\sigma$ of $\mathrm{Ord}^N_{n-|\sigma|+1}$ of size at most $8ns$. By Lemma 10, $\tilde{R}$ is of size at least

$$|\tilde{R}| \geq 2^{(n-|\sigma|)/2} \geq 2^{(n-2k)/2} \geq 2^{n/4},$$

therefore we obtain $8ns \geq 2^{n/4}$, and thus

$$s \geq 2^{n/4}/8n = 2^{n/4-\log n-3} = 2^{\Omega(n)}$$

which proves the claim. □

7 Implication graph formulas

In contrast to our result above, we now give an example where even the use of very small lemmas gives an exponential speed-up over tree-like resolution. We show that the implication graph formulas for every graph on n vertices have RTL(2)-refutations of linear size, whereas it is known that for some graphs they require exponential size tree-like resolution refutations [3].

Let a *pointed graph* be a directed acyclic graph with a unique sink t, where every vertex that is not a source has in-degree 2. The implication graph formula $\mathrm{Imp}(G)$ for such a pointed graph G consists of the source clause $x_s \vee y_s$ for every source s, the sink clauses $\bar{x}_t$ and $\bar{y}_t$, and the four implication clauses

$$\bar{x}_u \vee \bar{x}_v \vee x_w \vee y_w$$
$$\bar{x}_u \vee \bar{y}_v \vee x_w \vee y_w$$
$$\bar{y}_u \vee \bar{x}_v \vee x_w \vee y_w$$
$$\bar{y}_u \vee \bar{y}_v \vee x_w \vee y_w$$

for an inner vertex w with predecessors u and v.

Ben-Sasson et al. [3] show a lower bound for tree-like resolution refutations of the implication graph formulas for certain graphs:

Theorem 14. *There are pointed graphs G_n with n vertices such that tree-like resolution refutations of $\mathrm{Imp}(G_n)$ require size $2^{\Omega(n/\log n)}$.*

On the other hand, we have:

Theorem 15. *For every graph G with n vertices, there are RTL(2)-refutations of $\mathrm{Imp}(G)$ of size $O(n)$.*

Proof. For every vertex w with predecessors u and v, there is a tree-like derivation of $x_w \vee y_w$ from the lemmas $x_u \vee y_u$ and $x_v \vee y_v$ as follows:

First resolve $x_v \vee y_v$ with the first two implication clauses, giving $\bar{x}_u \vee x_w \vee y_w$. Also, resolve $x_v \vee y_v$ with the last two implication clauses to give $\bar{y}_u \vee x_w \vee y_w$. These two are resolved with $x_u \vee y_u$ to obtain $x_w \vee y_w$.

Now these derivations can be plugged together to yield an RTL(2)-derivation of $x_t \vee y_t$ from all the source clauses. Resolving this with the sink clauses gives the desired refutation. □

8 Conclusion

We have provided an example of a class of formulas which can be solved quickly by DLL algorithms with clause learning, but require exponential time when learning is restricted to short clauses. This rigorous lower bound result supports the experience made in practice that restricting to short clauses is not a useful heuristic for deciding which clauses to learn. The hard examples used are the formulas Ord_n based on the ordering principle, which frequently occur as hard examples in proof complexity.

It would be nice to have another example showing this behavior that has only short input clauses, but it seems likely that the technique of this paper can be extended to provide such an example, based on a 3-CNF extension of the formulas Ord_n or a restriction of Ord_n to the edges of an expander graph as used by Segerlind et al. [10]. This is being investigated in ongoing work.

A major problem is to extend the lower bounds to systems with lemmas of arbitrary length, and thus to separate the systems corresponding to DLL with clause learning [5, 7] – and thus the algorithm itself – from general dag-like resolution. For this problem, the techniques used here and in the earlier lower bound for the pigeonhole principle [5] are insufficient, since they rely heavily on the proofs being non-regular. But without the regularity restriction, the systems with arbitrary lemmas are equivalent to general resolution.

Acknowledgments. I thank Jan Hoffmann for helpful discussions about the results in this paper, and two reviewers for some useful suggestions.

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.

[2] 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.

[3] E. Ben-Sasson, R. Impagliazzo, and A. Wigderson. Near-optimal separation of general and tree-like resolution. *Combinatorica*, 24(4):585–604, 2004.

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

[5] 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.

[6] M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[7] 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.

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

[9] R. Letz. personal communication.

[10] 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.

[11] 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.

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

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

[14] A. van Gelder. Pool resolution and its relation to regular resolution and DPLL with clause learning. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR), LNAI 3835*, pages 580–594. Springer-Verlag, 2005.

[15] 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.