# On the Relative Efficiency of DPLL and OBDDs with Axiom and Join*

Matti Järvisalo

Department of Computer Science, University of Helsinki, Finland

**Abstract.** This paper studies the relative efficiency of ordered binary decision diagrams (OBDDs) and the Davis-Putnam-Logemann-Loveland procedure (DPLL), two of the main approaches to solving Boolean satisfiability instances. Especially, we show that OBDDs, even when constructed using only the rather weak axiom and join rules, can be exponentially more efficient than DPLL or, equivalently, tree-like resolution. Additionally, by strengthening via simple arguments a recent result—stating that such OBDDs do not polynomially simulate unrestricted resolution—we also show that the opposite holds: there are cases in which DPLL is exponentially more efficient out of the two considered systems. Hence DPLL and OBDDs constructed using only the axiom and join rules are polynomially incomparable. This further highlights differences between search-based and compilation-based approaches to Boolean satisfiability.

## 1 Introduction

Many algorithms for Boolean satisfiability (SAT) are based on either resolution (including most state-of-the-art search-based solvers today) or (reduced) ordered binary decision diagrams (OBDDs). Recently, there has been a lot of interest in the relative efficiency of satisfiability checking methods based on resolution and OBDDs [1,2,3,4,5,6]. While OBBDs in general are known to be in cases exponentially more efficient that unrestricted resolution [7], it has been recently shown [6] that the restricted $\mathsf{OBDD}_{\mathsf{aj}}$ proof system, consisting only of the rather weak *Axiom* and *Join* rules which correspond to the *Apply* OBDD operator (i.e., disallowing symbolic quantifier elimination and reordering), does not polynomially simulate unrestricted resolution. In other words, there is an infinite family $\{F_n\}_n$ of unsatisfiable CNF formulas such that (i) there is a polynomial-size resolution proof of $F_n$ w.r.t. $n$ for every $n$, whereas (ii) minimum-size $\mathsf{OBDD}_{\mathsf{aj}}$ proofs of $F_n$ for every $n$ are of exponential size w.r.t. $n$ (and of super-polynomial size w.r.t. the number of clauses in $F_n$). A practical implication of this result is that $\mathsf{OBDD}_{\mathsf{aj}}$ (under any variable ordering) does not polynomially simulate typical restarting conflict-driven clause learning SAT solvers—often the most efficient ones for practical applications of SAT, and which have been recently shown to polynomially simulate unrestricted resolution [8]. However, the results in [6] leave open the question of pinpointing the (in)efficiency of $\mathsf{OBDD}_{\mathsf{aj}}$ more exactly: Does it even polynomially simulate the Davis-Putnam-Logemann-Loveland procedure (DPLL) [9,10] that is known to be exponentially weaker than clause learning? Does DPLL polynomially simulate $\mathsf{OBDD}_{\mathsf{aj}}$?

* This work is financially supported by Academy of Finland (grant 132812).

In this paper we show that the answer to both of these questions is negative: 1. We show that DPLL (with an optimal branching heuristic) does not polynomially simulate $\mathsf{OBDD}_{\mathsf{aj}}$ (using a suitable variable ordering). 2. Strengthening the result of [6] via simple arguments, we show that the $\mathsf{OBDD}_{\mathsf{aj}}$ proof system (under any variable ordering) does not polynomially simulate DPLL (known to be equivalent to the tree-like resolution refinement). Hence $\mathsf{OBDD}_{\mathsf{aj}}$ and DPLL are *polynomially incomparable*.

**Theorem 1.** *OBDDs constructed using the Axiom and Join rules and* DPLL *(equivalently, tree-like resolution) are polynomially incomparable.*

This provides further understanding on the general question of the relative efficiency of DPLL and variants of OBDDs, highlighting further the differences between search-based and compilation-based approaches to Boolean satisfiability.

## 2 Preliminaries

**CNF Satisfiability.** A literal is a Boolean variable $x$ or its negation $\neg x$. A *clause* is a disjunction ($\vee$) of literals and a CNF formula is a conjunction ($\wedge$) of clauses. When convenient, we view a clause as a finite set of literals and/or a CNF formula as a finite set of clauses. The set of variables occurring in a CNF formula $F$ is denoted by $\mathsf{vars}(F)$, and the set of literals occurring in a clause $C$ by $\mathsf{lits}(C)$. An *assignment* $\tau$ is a function that maps literals to elements in $\{0, 1\}$, where 1 and 0, resp., stand for *true* and *false*, resp. If $\tau(x) = v$, then $\tau(\neg x) = 1 - v$, and vice versa. A clause is *satisfied* by $\tau$ if it contains at least one literal $l$ such that $\tau(l) = 1$. An assignment $\tau$ *satisfies* a CNF formula if it satisfies every clause in the formula.

**DPLL.** The DPLL procedure [9,10] is a classical complete search algorithm for deciding satisfiability of CNF formulas. It can be summarized as the following non-deterministic algorithm.

DPLL($F$)
  **If** $F$ is empty **report** *satisfiable* and **halt**
  **If** $F$ contains the empty clause **return**
  **Else** *choose a variable* $x \in \mathsf{vars}(F)$
    DPLL($F_x$)
    DPLL($F_{\neg x}$)

Here $F_x$ denotes the formula resulting from applying *unit propagation* until fixpoint on $F$, i.e., removing all clauses containing $x$ and all occurrences of $\neg x$ from $F$, and repeating until fixpoint for all single-literal clauses in $F$. Practical implementations make DPLL deterministic by implementing a branching heuristic for choosing a variable. However, here we do not restrict this non-deterministic choice. A DPLL proof of an unsatisfiable CNF formula $F$ is a search tree of DPLL($F$). The size of a DPLL proof is the number of nodes in the tree.

**Resolution.** The well-known Resolution proof system (RES) is based on the *resolution rule* that allows one to *directly derive* the clause $(C \cup D) \setminus \{x, \neg x\}$ from the clauses $\{x\} \cup C$ and $\{\neg x\} \cup D$ by *resolving on* the variable $x$. Given an unsatisfiable CNF

formula $F$, a RES *proof* of $F$ is a sequence of clauses $\pi = (C_1, C_2, \ldots, C_m = \emptyset)$, where each $C_i$, $1 \leq i \leq m$, is either (i) a clause in $F$ (an *initial clause*) or (ii) directly derived with the resolution rule from two clauses $C_j, C_k$ where $1 \leq j, k < i$. The *size* of $\pi$, denoted by $|\pi|$, is $m$. Any RES proof $\pi = (C_1, C_2, \ldots, C_m = \emptyset)$ can be presented as a directed acyclic graph. The clauses occurring in $\pi$ label the nodes. The edge relation is defined so that there are edges from $C_i$ and $C_j$ to $C_k$, if and only if $C_k$ has been directly derived from $C_i$ and $C_j$. *Tree-like Resolution* (T-RES) proofs are representable as trees. It is well-known that T-RES proofs are polynomially equivalent to search trees traversed by the DPLL procedure.

In *Extended Resolution* (E-RES) [11] one can first apply the *extension rule* to add a conjunction of clauses (an *extension*) to a CNF formula $F$ in a restricted manner, before using the resolution rule to construct a RES proof of the resulting formula. In more detail, for a given CNF formula $F$, the extension rule allows for iteratively adding definitions of the form $x \equiv l_1 \wedge l_2$ (i.e. the clauses $(x \vee \neg l_1 \vee \neg l_2)$, $(\neg x \vee l_1)$, and $(\neg x \vee l_2)$) to $F$, where $x$ is a new variable and $l_1$, $l_2$ are literals in the current formula. The resulting formula $F \wedge E$ then consists of the original formula $F$ and the extension $E$, the conjunction of the clauses iteratively added to $F$ using the extension rule.

**OBDDs with Axiom and Join.** A *binary decision diagram* (BDD) over a set of Boolean variables $V$ is a rooted directed acyclic graph that consists of (i) *decision nodes* labelled with distinct variables from $V$ and (ii) two terminal nodes (of out-degree zero) labelled with 0 and 1. Each decision node $v$ has two children, low($v$) and high($v$). The edge from $v$ to low($v$) (to high($v$), resp.) represents assigning $v$ to 0 (to 1, resp.). A BDD is *ordered* according to a total variable order $\prec$ if its variables appear in the order given by $\prec$ on all paths from the root to the terminal nodes. An ordered BDD is *reduced* (simply, an OBDD from here on) if its (i) isomorphic subgraphs have been merged, and (ii) the nodes that have isomorphic children have been eliminated. Given any propositional formula $\phi$ and a total variable order $\prec$ over $\mathsf{vars}(\phi)$, there is a unique OBDD $\mathsf{B}(\phi, \prec)$ that represents $\phi$. The *size* of $\mathsf{B}(\phi, \prec)$, denoted by $\mathsf{size}(\mathsf{B}_i(\phi_i, \prec))$, is the number of its nodes.

Given an unsatisfiable CNF formula $F$ and a total variable order $\prec$ over $\mathsf{vars}(F)$, an $\mathsf{OBDD}_{\mathsf{aj}}$ *proof* of $F$ (i.e., an $\mathsf{OBDD}_{\mathsf{aj}}$ *derivation* of the OBDD for 0) is a sequence $\rho = (\mathsf{B}_1(\phi_1, \prec), \ldots, \mathsf{B}_m(\phi_m, \prec))$ of OBDDs, where (i) $\mathsf{B}_m(\phi_m, \prec)$ is the single-node OBDD representing 0, and (ii) for each $i \in \{1, \ldots, m\}$, either

- $\phi_i$ is a clause in $F$, or
- $\phi_i = \phi_j \wedge \phi_k$ for some $\mathsf{B}_j(\phi_j, \prec)$ and $\mathsf{B}_k(\phi_k, \prec)$, $1 \leq j < k < i$, in $\rho$.

In the former case $\mathsf{B}_i(\phi_i, \prec)$ is an *axiom*, and in the latter $\mathsf{B}_i(\phi_i, \prec)$ is the *join* of $\mathsf{B}_j(\phi_j, \prec)$ and $\mathsf{B}_k(\phi_k, \prec)$. The size of an $\mathsf{OBDD}_{\mathsf{aj}}$ proof $\rho$ is $\Sigma_{i=1}^{m} \mathsf{size}(\mathsf{B}_i(\phi_i, \prec))$.

## 3 DPLL does not Polynomially Simulate $\mathsf{OBDD}_{\mathsf{aj}}$

In this section we show that DPLL does not polynomially simulate $\mathsf{OBDD}_{\mathsf{aj}}$. For the separation, we consider so-called *pebbling contradictions* $\mathrm{Peb}(G)$, first introduced in [12], based on the structure of a directed acyclic graph (DAG) $G$. Taking two variables $x_{i,0}$ and $x_{i,1}$ for each node in $G$, $\mathrm{Peb}(G)$ is the conjunction of the following clauses.

– $(x_{i,0} \vee x_{i,1})$ for each source node (in-degree 0) $i$ of $G$;
– $(\neg x_{i,0})$ and $(\neg x_{i,1})$ for each sink node (out-degree 0) $i$ of $G$;
– $(\neg x_{i_1,a_1} \vee \cdots \vee \neg x_{i_k,a_k} \vee x_{j,0} \vee x_{j,1})$ for each non-source node $j$, where $i_1, \ldots, i_k$ are the predecessors of $j$, and for each $(a_1, \ldots, a_k) \in \{0,1\}^k$.

The following theorem helps us in achieving polynomial-size $\mathsf{OBDD}_{\mathsf{aj}}$ proofs.

**Theorem 2 ([13]).** *For any Boolean function $f$ over $n$ variables, and any variable order $\prec$, $\mathsf{size}(\mathsf{B}(f, \prec)) = \mathcal{O}(2^n/n)$.*

**Corollary 1.** *For any unsatisfiable CNF formula $F$ over $n$ variables, and any variable order $\prec$, there is an $\mathsf{OBDD}_{\mathsf{aj}}$ proof of $F$ of size $2^{\mathcal{O}(n)}$.*

The following two lemmas play a key role in this work. For proving the lemmas we rely on a similar proof strategy as the one applied in [14] used in a different context (for showing that tree-like resolution does not polynomially simulate *ordered resolution*).

**Lemma 1.** *Let $G$ be a DAG on $n$ nodes, and $j$ a node in $G$ with parents $i_1, \ldots, i_k$ where $k = \mathcal{O}(\log n)$. Consider the clauses $(x_{i_1,0} \vee x_{i_1,1})$,…, $(x_{i_k,0} \vee x_{i_k,1})$ and $(\neg x_{i_1,a_1} \vee \cdots \vee \neg x_{i_k,a_k} \vee x_{j,0} \vee x_{j,1})$ for all $(a_1, \ldots, a_k) \in \{0,1\}^k$. For any variable order $\prec$, there is a polynomial-size $\mathsf{OBDD}_{\mathsf{aj}}$ derivation of $\mathsf{B}((x_{j,0} \vee x_{j,1}), \prec)$ from these clauses.*

*Proof.* Consider the unsatisfiable CNF formula consisting of the clauses $(x_{i_1,0} \vee x_{i_1,1})$,…, $(x_{i_k,0} \vee x_{i_k,1})$ and $(\neg x_{i_1,a_1} \vee \cdots \vee \neg x_{i_k,a_k})$ for all $(a_1, \ldots, a_k) \in \{0,1\}^k$. The number of variables in this formula is $\mathcal{O}(\log n)$, and hence by Corollary 1 there is a polynomial-size $\mathsf{OBDD}_{\mathsf{aj}}$ proof of this formula for any variable order $\prec$. Such an $\mathsf{OBDD}_{\mathsf{aj}}$ proof can be transformed into a $\mathsf{OBDD}_{\mathsf{aj}}$ derivation of $\mathsf{B}((x_{j,0} \vee x_{j,1}), \prec')$ by defining $\prec'$ as $\prec$ to which $x_{j,0}$ and $x_{j,1}$ have been added as the last two elements, and by replacing $\mathsf{B}(\phi, \prec)$ with $\mathsf{B}(\phi \vee x_{j,0} \vee x_{j,1}, \prec)$ for each $\mathsf{B}(\phi, \prec)$ in the proof such that either $\phi$ is $(\neg x_{i_1,a_1} \vee \cdots \vee \neg x_{i_k,a_k})$ or $\mathsf{B}(\phi, \prec)$ has been derived starting from the axiom $\mathsf{B}((\neg x_{i_1,a_1} \vee \cdots \vee \neg x_{i_k,a_k}), \prec)$. For each such $\mathsf{B}(\phi, \prec)$, $\mathsf{B}(\phi \vee x_{j,0} \vee x_{j,1}, \prec)$ is $\mathsf{B}(\phi, \prec)$ with the terminal node 0 replaced by $\mathsf{B}((x_{j,0} \vee x_{j,1}), \prec)$. □

**Lemma 2.** *There are polynomial-size $\mathsf{OBDD}_{\mathsf{aj}}$ proofs of $\mathrm{Peb}(G)$ for any DAG $G$ with node in-degree bounded by $\mathcal{O}(\log n)$.*

*Proof.* Fix any ordering $\prec$ of the variables in $\mathrm{Peb}(G)$ that respects a topological ordering of $G$. Label each source $j$ of $G$ with $\mathsf{B}((x_{j,0} \vee x_{j,1}), \prec)$. For each non-source node $j$ of $G$ with parents $i_1, \ldots, i_k$, $k = \mathcal{O}(\log n)$, replace $j$ with the polynomial-size $\mathsf{OBDD}_{\mathsf{aj}}$ derivation of $\mathsf{B}((x_{j,0} \vee x_{j,1}), \prec)$ (Lemma 1) under $\prec$. The result is a polynomial-size $\mathsf{OBDD}_{\mathsf{aj}}$ derivation of $\mathsf{B}((x_{t,0} \vee x_{t,1}), \prec)$ for the single sink $t$ of $G$ (analogously for multiple sinks). To complete the proof, join $\mathsf{B}((x_{t,0} \vee x_{t,1}), \prec)$ with the axioms $\mathsf{B}((\neg x_{t,0}), \prec)$ and $\mathsf{B}((\neg x_{t,1}), \prec)$. □

Combined with the following lemma, we have that DPLL (equivalently, T-RES) does not polynomially simulate $\mathsf{OBDD}_{\mathsf{aj}}$ (using a suitable variable order for $\mathsf{OBDD}_{\mathsf{aj}}$).

**Lemma 3 ([12]).** *There is an infinite family $\{G_n\}$ of DAGs with constant node in-degree (from [15]) such that minimum-size T-RES proofs of $\mathrm{Peb}(G_n)$ are of size $2^{\Omega(n/\log n)}$.*

## 4 $\mathsf{OBDD}_{\mathsf{aj}}$ does not Polynomially Simulate DPLL

In [6] it was shown that $\mathsf{OBDD}_{\mathsf{aj}}$ does not polynomially simulate unrestricted resolution RES. In this section we show the stronger result that $\mathsf{OBDD}_{\mathsf{aj}}$ is not only weaker than RES, but also exponentially weaker than DPLL (equivalently, T-RES).

### 4.1 $\mathsf{OBDD}_{\mathsf{aj}}$ does not Benefit from the Extension Rule

As an auxiliary result, we prove the following lemma as an extension of [6, Lemma 8]. The original lemma was restricted to a particular CNF formula $\mathrm{PHP}_n^{n+1}$ and a particular extension of $\mathrm{PHP}_n^{n+1}$. This more general version states that $\mathsf{OBDD}_{\mathsf{aj}}$ proofs cannot be made smaller by first adding an extension to the input unsatisfiable CNF formula.

**Lemma 4.** *Assume an arbitrary unsatisfiable CNF formula $F$ and extension $E$ to $F$, and any satisfiable $F' \subset F$ and $E' \subseteq E$. Then, for every variable order $\prec$ over $\mathsf{vars}(F') \cup \mathsf{vars}(E')$, $\mathsf{size}(\mathsf{B}(F' \wedge E', \prec)) \geq \mathsf{size}(\mathsf{B}(F', \prec))$.*

Following the proof strategy for [6, Lemma 8], we first state a simple extension of [6, Lemma 7], simply stating that no extension $E$ of a CNF formula $F$ can affect the set of satisfying assignments of $F$ (restricted to $F$).

**Lemma 5.** *Assume an arbitrary CNF formula $F$ and extension $E$ to $F$, any satisfiable $F' \subset F$ and $E' \subseteq E$, and an assignment $\tau$ that satisfies $F'$. Then there is an assignment $\tau'$ such that (i) $\tau'(x) = \tau(x)$ for each $x \in \mathsf{vars}(F')$, and (ii) $\tau'$ satisfies $F' \wedge E'$.*

*Proof.* Assume that the clauses in $E = C_1 \wedge \cdots \wedge C_k$ were introduced using the extension rule in the order of the sequence $(C_1, \ldots, C_k)$. Fix an arbitrary satisfiable $F' \subseteq F$ and assignment $\tau$ that satisfies $F'$. Let $\tau'(x) = \tau(x)$ for each $x \in \mathsf{vars}(F')$. By induction, assume that, for an arbitrary $i$, $\tau'$ satisfies all $C_j$ for $j < i$. Let $C_i$ be part of a definition $x_i \equiv l \wedge l'$. To satisfy $C_i$, we extend $\tau'$ as follows. If both $l$ and $l'$ are assigned under $\tau'$, then assign $x_i$ so that the semantics of $x_i \equiv l \wedge l'$ is respected. If $l$ (or $l'$, resp.) is not assigned under $\tau'$ (this is possible in case $l$ or $l'$ do not appear in $F'$), first assign it an arbitrary value. □

For the following, a function $f$ *depends essentially* on a variable $x$ if $f|_{x=0} \neq f|_{x=1}$, where $f|_{x=c}$ denotes the function $f$ with $x$ assigned to a constant $c \in \{0, 1\}$. Again following [6], we make use of a structural theorem from [16].

**Theorem 3 ([16]).** *For any Boolean function $f(x_1, \ldots, x_n)$ and $i < n$, let $S_i$ be the set $\{f|_{x_1=c_1,\ldots,x_{i-1}=c_{i-1}} : c_1, \ldots, c_{i-1} \in \{0, 1\}\}$ of sub-functions which depend essentially on $x_i$. Then the OBDD for $f$ under the variable order $x_1 \prec \cdots \prec x_n$ contains exactly $|S_i|$ nodes labelled with $x_i$ in correspondence with the sub-functions in $S_i$.*

In the following, for an assignment $\tau$ over a set $X$ of variables and a variable order $\prec$ over $V$, let $\tau_{\prec x}$ be the restriction of $\tau$ to the variables preceding $x \in X$ under $\prec$.

*Proof of Lemma 4.* We show that, for any $F', E', \prec, i < |\mathsf{vars}(F')|$, and $x_i \in \mathsf{vars}(F')$, where $x_i$ is the $i$th variable in $\mathsf{vars}(F')$ under $\prec$, it holds that if $\mathsf{B}(F', \prec)$ has $k$ nodes labelled with $x_i$, then $\mathsf{B}(F' \wedge E', \prec)$ has at least $k$ nodes labelled with $x_i$.

Take any satisfying assignment $\tau$ over $\mathsf{vars}(F')$ such that $F'|_{\tau_{\prec x_i}}$ depends essentially on $x_i$. By Theorem 3, corresponding to any such $F'|_{\tau_{\prec x_i}}$ there is a node $n_{\tau_{\prec x_i}}$ in $\mathsf{B}(F', \prec)$ labelled with $x_i$. Based on $\tau$, consider an assignment $\tau'$ for $F' \wedge E'$ as in Lemma 5. By the construction of $\tau'$, $(F' \wedge E')|_{\tau'_{\prec x_i}}$ depends essentially on $x_i$. By Theorem 3, for any such $n_{\tau_{\prec x_i}}$, there is a distinct node $n_{\tau'_{\prec x_i}}$ (corresponding to $(F' \wedge E')|_{\tau'_{\prec x_i}}$) in $\mathsf{B}(F' \wedge E', \prec)$. □

The following is an immediately corollary of Lemma 4.

**Corollary 2.** *Let $F$ be an unsatisfiable CNF formula and $E$ an extension to $F$. For any variable order $\prec$ over $\mathsf{vars}(F) \cup \mathsf{vars}(E)$, if $F \wedge E$ has a $\mathsf{OBDD}_{\mathsf{aj}}$ proof of size $s$, then $F$ has a $\mathsf{OBDD}_{\mathsf{aj}}$ proof of size $s$.*

### 4.2 DPLL and the Extension Rule

Let $F$ be an arbitrary unsatisfiable CNF formula and let $\pi_F = (C_1, \ldots, C_m = \emptyset)$ be a RES proof of $F$. We define the extension $\mathrm{E}(\pi_F)$ of $F$ based on $\pi_F$, defining new variables $e_i \equiv C_i$ for $i = 1, \ldots, m-1$ using the extension rule, as the CNF formula

$$\mathrm{E}(\pi_F) := \bigwedge_{i=1}^{m-1} \Big( (\neg e_i \vee C_i) \wedge \bigwedge_{l \in \mathsf{lits}(C_i)} (e_i \vee \neg l) \Big).$$

This formulation originates from a construction that was used to polynomially simulate Frege systems by tree-like Frege systems [17], and was also applied in [18].

**Lemma 6.** *Let $F$ be an unsatisfiable CNF formula and let $\pi_F$ be a RES proof of $F$. There is a DPLL proof of $F \wedge \mathrm{E}(\pi_F)$ of size $\mathcal{O}(|\pi_F|)$.*

*Proof.* Choose variables in the order $e_1 \prec \cdots \prec e_{m-1}$. For each $i = 1, \ldots, m-1$, the call DPLL($F \wedge \mathrm{E}(\pi_F)_{e_1,\ldots,e_{i-1},\neg e_i}$) returns immediately since $F \wedge \mathrm{E}(\pi_F)_{e_1,\ldots,e_{i-1},\neg e_i}$ contains the empty clause due to emptying either a clause in $F$, or one of the two clauses in $\pi_F$ used to directly derive the resolvent $C_i$. The call DPLL($F \wedge \mathrm{E}(\pi_F)_{e_1,\ldots,e_{m-1}}$) returns immediately since there are the two unit clauses $(x)$ and $(\neg x)$ in $\pi_F$ for some variable $x$. □

In fact, as similarly observed in [18], full one-step lookahead with unit propagation is enough for constructing the DPLL proof presented in the proof of Lemma 6.

### 4.3 Separating DPLL from $\mathsf{OBDD}_{\mathsf{aj}}$

The well-known *pigeon-hole principle* states that there is no injective mapping from an $m$-element set into an $n$-element set if $m > n$ (that is, $m$ pigeons cannot sit in fewer than $m$ holes so that every pigeon has its own hole). We will consider the case $m = n + 1$ encoded as the CNF formula

$$\mathrm{PHP}_n^{n+1} := \bigwedge_{i=1}^{n+1} \Big( \bigvee_{j=1}^{n} p_{i,j} \Big) \wedge \bigwedge_{j=1}^{n} \bigwedge_{i=1}^{n} \bigwedge_{i'=i+1}^{n+1} (\neg p_{i,j} \vee \neg p_{i',j}),$$

where each $p_{i,j}$ is a Boolean variable with the interpretation "$p_{i,j}$ is 1 if and only if the $i^{\mathrm{th}}$ pigeon sits in the $j^{\mathrm{th}}$ hole". Notice that $\mathrm{PHP}_n^{n+1}$ contains $O(n^2)$ clauses.

**Theorem 4 ([19]).** *There is no polynomial-size* RES *proof of* $\mathrm{PHP}_n^{n+1}$.

In contrast, Cook [20] showed that there is a polynomial-size E-RES proof of $\mathrm{PHP}_n^{n+1}$. Cook basically applies the E-RES extension rule to add a conjunction $\mathrm{EXT}_n$ of $O(n^3)$ clauses to $\mathrm{PHP}_n^{n+1}$, based on defining new variables $p_{i,i}^k \equiv p_{i,j}^{k-1} \vee (p_{i,n}^{k-1} \wedge p_{n+1,j}^{k-1})$, where $1 \leq i \leq n$, $1 \leq j \leq n-1$, $1 \leq k \leq n-1$, and each $p_{i,j}^0$ is the variable $p_{i,j} \in \mathsf{vars}(\mathrm{PHP}_n^{n+1})$. These equivalences[1] are presented as the CNF formula
$\mathrm{EXT}_n := \bigwedge_{k=1}^{n-1} \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n-1}$
$\Big( \big(p_{i,j}^k \vee \neg p_{i,j}^{k-1}\big) \wedge \big(p_{i,j}^k \vee \neg p_{i,n}^{k-1} \vee \neg p_{n+1,j}^{k-1}\big) \wedge \big(\neg p_{i,j}^k \vee p_{i,j}^{k-1} \vee p_{i,n}^{k-1}\big) \wedge \big(\neg p_{i,j}^k \vee p_{i,j}^{k-1} \vee p_{n+1,j}^{k-1}\big) \Big)$.

**Theorem 5 ([20]).** *There is a* RES *proof of* $\mathrm{PHP}_n^{n+1} \wedge \mathrm{EXT}_n$ *of size* $\mathcal{O}(n^4)$.

Explicit constructions of a RES proof of size $\mathcal{O}(n^4)$ of $\mathrm{PHP}_n^{n+1} \wedge \mathrm{EXT}_n$ are presented in [18,6]. On the other hand, these proofs are not tree-like, and it is not apparent whether there is a polynomial-size DPLL proof of $\mathrm{PHP}_n^{n+1} \wedge \mathrm{EXT}_n$. However, we can use the extension trick from Sect. 4.2 for achieving a short DPLL proof.

**Corollary 3.** *There is an extension* $E$ *to* $\mathrm{PHP}_n^{n+1}$ *such that there is a polynomial-size* DPLL *proof of* $\mathrm{PHP}_n^{n+1} \wedge E$.

*Proof.* Take an arbitrary RES proof $\pi$ of $\mathrm{PHP}_n^{n+1} \wedge \mathrm{EXT}_n$ such that $|\pi| \in \mathcal{O}(n^4)$ (there is such a $\pi$ by Theorem 5). Define $E$ as $\mathrm{EXT}_n \wedge \mathrm{E}(\pi)$. By Lemma 6 there is a polynomial-size DPLL proof of $\mathrm{PHP}_n^{n+1} \wedge \mathrm{EXT}_n \wedge \mathrm{E}(\pi)$. □

To separate DPLL from $\mathsf{OBDD}_{\mathsf{aj}}$, we observe the following.

**Theorem 6 ([6]).** *For any variable order* $\prec$, *minimum-size* $\mathsf{OBDD}_{\mathsf{aj}}$ *proofs of* $\mathrm{PHP}_n^{n+1}$ *are of size* $2^{\Omega(n)}$.

**Corollary 4.** *Let* $E$ *be an arbitrary extension of* $\mathrm{PHP}_n^{n+1}$. *For any variable order* $\prec$, *minimum-size* $\mathsf{OBDD}_{\mathsf{aj}}$ *proofs of* $\mathrm{PHP}_n^{n+1} \wedge E$ *are of size* $2^{\Omega(n)}$.

*Proof.* Follows directly from Corollary 2 and Theorem 6. □

The fact that $\mathsf{OBDD}_{\mathsf{aj}}$ does not polynomially simulate DPLL (equivalently, T-RES) now follows directly from Corollaries 3 and 4.

## 5 Conclusions

We showed that the standard DPLL procedure and OBDDs constructed using the *axiom* and *join* rules ($\mathsf{OBDD}_{\mathsf{aj}}$) are polynomially incomparable. This further highlights the differences between search-based and compilation-based approaches to Boolean satisfiability. Especially, although $\mathsf{OBDD}_{\mathsf{aj}}$ is intuitively rather weak, it can still be exponentially more efficient than DPLL. However, in contrast to DPLL, $\mathsf{OBDD}_{\mathsf{aj}}$ cannot

[1] Although Cook introduces directly clauses representing $p_{i,i}^k \equiv p_{i,j}^{k-1} \vee (p_{i,n}^{k-1} \wedge p_{n+1,j}^{k-1})$ which does not follow the original definition of the extension rule, it is easy to see that this can be simulated with the original rule by first introducing an auxiliary variable for $(p_{i,n}^{k-1} \wedge p_{n+1,j}^{k-1})$. This more general way of defining the extension does not affect the results of this paper.

exploit particular types of redundancy in CNF (introduced by the extension rule). As a result, DPLL can be in cases exponentially more efficient than $\mathsf{OBDD}_{\mathsf{aj}}$.

Whether there is a resolution refinement that polynomially simulate $\mathsf{OBDD}_{\mathsf{aj}}$ is an interesting open question. Another interesting question is the relative efficiency of tree-like and DAG-like $\mathsf{OBDD}_{\mathsf{aj}}$ proofs. Especially, the $\mathsf{OBDD}_{\mathsf{aj}}$ proofs constructed in Lemma 2 are not tree-like.

## References


1. Groote, J.F., Zantema, H.: Resolution and binary decision diagrams cannot simulate each other polynomially. Discrete Applied Mathematics **130**(2) (2003) 157–171
2. Atserias, A., Kolaitis, P.G., Vardi, M.Y.: Constraint propagation as a proof system. In: Proc. CP. Volume 3258 of Lecture Notes in Computer Science., Springer (2004) 77–91
3. Sinz, C., Biere, A.: Extended resolution proofs for conjoining BDDs. In: Proc. CSR. Volume 3967 of Lecture Notes in Computer Science., Springer (2006) 600–611
4. Segerlind, N.: On the relative efficiency of resolution-like proofs and ordered binary decision diagram proofs. In: Proc. CCC, IEEE Computer Society (2008) 100–111
5. Peltier, N.: Extended resolution simulates binary decision diagrams. Discrete Applied Mathematics **156**(6) (2008) 825–837
6. Tveretina, O., Sinz, C., Zantema, H.: Ordered binary decision diagrams, pigeonhole formulas and beyond. Journal on Satisfiability, Boolean Modeling and Computation **7** (2010) 35–58
7. Chen, W., Zhang, W.: A direct construction of polynomial-size OBDD proof of pigeon hole problem. Information Processing Letters **109**(10) (2009) 472—477
8. Pipatsrisawat, K., Darwiche, A.: On the power of clause-learning SAT solvers as resolution engines. Artificial Intelligence **175**(2) (2011) 512–525
9. Davis, M., Putnam, H.: A computing procedure for quantification theory. Journal of the ACM **7**(3) (1960) 201–215
10. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem proving. Communications of the ACM **5**(7) (1962) 394–397
11. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In Slisenko, A., ed.: Studies in Constructive Mathematics and Mathematical Logic, Part II. Volume 8 of Seminars in Mathematics, V.A. Steklov Mathematical Institute, Leningrad. Consultants Bureau (1969) 115–125 (originally in Russian).
12. Ben-Sasson, E., Wigderson, A.: Short proofs are narrow - resolution made simple. Journal of the ACM **48**(2) (2001) 149–169
13. Liaw, H.T., Lin, C.S.: On the OBDD-representation of general boolean functions. IEEE Transactions on Computers **41**(6) (1992) 661–664
14. Buresh-Oppenheim, J., Pitassi, T.: The complexity of resolution refinements. Journal of Symbolic Logic **72**(4) (2007) 1336–1352
15. Paul, W.J., Tarjan, R.E., Celoni, J.R.: Space bounds for a game on graphs. Mathematical Systems Theory **10** (1977) 239–251
16. Sieling, D., Wegener, I.: NC-algorithms for operations on binary decision diagrams. Parallel Processing Letters **3** (1993) 3–12
17. Krajicek, J.: Speed-up for propositional frege systems via generalizations of proofs. Commentationes Mathematicae Universitas Carolinae **30**(1) (1989) 137–140
18. Järvisalo, M., Junttila, T.: Limitations of restricted branching in clause learning. Constraints **14**(3) (2009) 325–356
19. Haken, A.: The intractability of resolution. Theoretical Computer Science **39**(2–3) (1985) 297–308
20. Cook, S.A.: A short proof of the pigeon hole principle using extended resolution. SIGACT News **8**(4) (1976) 28–32