DIRECTIONAL RESOLUTION: THE DAVIS-PUTNAM PROCEDURE, REVISITED

Rina Dechter
Department of Information and Computer Science
University of California, Irvine

Irina Rish
Department of Information and Computer Science
University of California, Irvine

Technical Report 94-54
November 22, 1994

Abstract

The paper presents algorithm *directional resolution*, a variation on the original Davis-Putnam algorithm, and analyzes its worst-case behavior as a function of the topological structure of the theories. The notions of *induced width* and *diversity* are shown to play a key role in bounding the complexity of the procedure. The importance of our analysis lies in highlighting structure-based tractable classes of satisfiability and in providing theoretical guarantees on the time and space complexity of the algorithm. Contrary to previous assessments, we show that for many theories directional resolution could be an effective procedure. Our empirical tests confirm theoretical prediction, showing that on problems with special structures, like *chains*, directional resolution greatly outperforms one of the most effective satisfiability algorithm known to date, namely the popular Davis-Putnam procedure.

To be published at KR-94

Directional Resolution: The Davis-Putnam Procedure, Revisited [*]

Rina Dechter
Information and Computer Science
University of California, Irvine
dechter@ics.uci.edu

Irina Rish
Information and Computer Science
University of California, Irvine
irinar@ics.uci.edu

Abstract

The paper presents algorithm *directional resolution*, a variation on the original Davis-Putnam algorithm, and analyzes its worst-case behavior as a function of the topological structure of the theories. The notions of *induced width* and *diversity* are shown to play a key role in bounding the complexity of the procedure. The importance of our analysis lies in highlighting structure-based tractable classes of satisfiability and in providing theoretical guarantees on the time and space complexity of the algorithm. Contrary to previous assessments, we show that for many theories directional resolution could be an effective procedure. Our empirical tests confirm theoretical prediction, showing that on problems with special structures, like *chains*, directional resolution greatly outperforms one of the most effective satisfiability algorithm known to date, namely the popular Davis-Putnam procedure.

1 Introduction

In 1960, Davis and Putnam [Davis and Putnam, 1960] presented their resolution algorithm. They proved that a restricted amount of resolution, if performed systematically along some order of the atomic formulas, is sufficient for deciding satisfiability. This algorithm, in its original form, has received limited attention, and analyses of its performance have emphasized its worst-case exponential behavior [Galil, 1977, Goerdt, 1992], while neglecting its virtues. This happened, in our view, because the algorithm was immediately overshadowed by a competitor with nearly the same name: *The Davis-Putnam Procedure*. This competing algorithm, proposed in 1962 by Davis, Logemann, and Loveland [Davis et al., 1962], searches through the space of possible truth assignments while performing unit resolution until quiesience at each step. We will refer to the first algorithm as $DP-elimination$ and to the second as $DP-backtracking$. The latter was presented in [Davis et al., 1962] as a minor syntactic change to the first: the *elimination rule* (rule III in [Davis and Putnam, 1960]) in DP-elimination was replaced by the *splitting rule* (rule III' in [Davis et al., 1962]) in order to avoid the memory explosion encountered when empirically testing DP-elimination. By refraining from an explicit analysis of this exchange (beyond the short comment on memory explosion), the authors of [Davis et al., 1962] may have left the impression that the two algorithms are basically identical. Indeed, from then on, most work on the Davis-Putnam procedure quotes the backtracking version [Goldberg et al., 1982, Selman, 1992], wrongly suggesting that this is the algorithm presented in [Davis and Putnam, 1960].

In this paper, we wish to "revive" the DP-elimination algorithm by studying its virtues theoretically and by subjecting it to a more extensive empirical testing. First, we show that, in addition to determining satisfiability, the algorithm generates an equivalent theory that facilitates model generation and query processing. Consequently, it may be better viewed as a knowledge compilation algorithm. Second, we offset the known worst-case exponential complexities [Galil, 1977, Goerdt, 1992] by showing that the algorithm is tractable for many of the known tractable classes for satisfiability (e.g., 2-*cnfs* and Horn clauses) and for constraint satisfaction problems [Dechter and Pearl, 1987, Dechter and Pearl, 1991] (e.g., causal theories and theories having a bounded induced width). Third, we present a new parameter, called *diversity*, that gives rise to new tractable classes.

On the empirical side, we qualify prior empirical tests in [Davis et al., 1962] by showing that for uniform random propositional theories DP-backtracking outperforms DP-elimination by far. However, for a

[*]This work was partially supported by NSF grant IRI-9157636, by Air Force Office of Scientific Research grant AFOSR 900136, by Toshiba of America, and by a Xerox grant.

class of instances having a *chain*-like structure DP-elimination outperforms DP-backtracking by several orders of magnitude.

2 Definition and preliminaries

We denote propositional symbols, also called *variables*, by uppercase letters $P, Q, R, \ldots$, propositional literals (i.e., $P, \neg P$) by lowercase letters $p, q, r, \ldots$, and disjunctions of literals, or *clauses*, by $\alpha, \beta, \ldots$. For instance, $\alpha = (P \vee Q \vee R)$ is a clause. We will sometime denote by $\{P, Q, R\}$ the clause $(P \vee Q \vee R)$. A *unit clause* is a clause of size 1. The notation $(\alpha \vee T)$ will be used as shorthand for the disjunction $(P \vee Q \vee R \vee T)$, and $\alpha \vee \beta$ denotes the clause whose literal appears in either α or β. The *resolution* operation over two clauses $(\alpha \vee Q)$ and $(\beta \vee \neg Q)$ results in a clause $(\alpha \vee \beta)$, thus eliminating Q. *Unit resolution* is a resolution operation when one of the clauses is a unit clause. A formula φ in conjunctive normal form (*cnf*) is a set of clauses $\varphi = \{\alpha_1, \ldots, \alpha_t\}$ that denotes their conjunction. The set of *models* of a formula φ is the set of all satisfying truth assignments to all its symbols. A clause α is *entailed* by φ, $\varphi \models \alpha$, iff α is true in all models of φ. A Horn formula is a cnf formula whose clauses all have at most one positive literal. A definite formula is a cnf formula that has exactly one positive literal. A clause is positive if it contains only positive literals and is negative if it contains negative literals only. A *k-cnf* formula is one whose clauses are all of length k or less.

3 DP-elimination – Directional Resolution

The DP-elimination [Davis and Putnam, 1960] is an ordering-based restricted resolution that can be described as follows. Given an arbitrary ordering of the propositional variables, we assign to each clause the index of the highest ordered literal in that clause. Then we resolve only clauses having the same index, and only on their highest literal. The result of this restriction is a systematic elimination of literals from the set of clauses that are candidates for future resolution. DP-elimination also includes additional steps, one forcing unit resolution whenever possible and another preferring resolution over literals that appear only negatively (called *all-negative*) or only positively (called *all-positive*). There are many other intermediate steps that can be introduced between the basic steps of eliminating the highest indexed variable (i.e., subsumption elimination). However, in this paper, we will focus on the ordered elimination step and will invoke auxiliary steps whenever necessary. Additionally, we will be interested not merely in achieving refutation, but also in the sum total of the clauses accumulated by this process, which constitutes an equivalent theory with useful computational features. Algorithm *directional resolution* (*DR*) (the core of DP-elimination) is described in Figure 1. We call its output theory, $E_d(\varphi)$, the *directional extension* of φ.

```
directional-resolution
Input: A cnf theory $\varphi$, an ordering $d = Q_1, \ldots, Q_n$ of its variables.
Output: A decision of whether $\varphi$ is satisfiable. If it is, a theory $E_d(\varphi)$, equivalent to $\varphi$, else an empty directional extension.
1. Initialize: generate an ordered partition of the clauses, $bucket_1, \ldots, bucket_n$, where $bucket_i$ contains all the clauses whose highest literal is $Q_i$.
2. For $i = n$ to 1 do:
3. Resolve each pair $\{(\alpha \vee Q_i), (\beta \vee \neg Q_i)\} \subseteq bucket_i$. If $\gamma = \alpha \vee \beta$ is empty, return $E_d(\varphi) = \emptyset$, the theory is not satisfiable; else, determine the index of $\gamma$ and add it to the appropriate bucket.
4. End-for.
5. Return $E_d(\varphi) \Longleftarrow \bigcup_i bucket_i$.
```

Figure 1: Algorithm *directional resolution*

The algorithm can be conveniently described using a partitioning of the set of clauses of a theory into buckets. Given an ordering $d = Q_1, \ldots Q_n$, the bucket for Q_i $bucket_i$, contains all the clauses containing Q_i that do not contain any symbol higher in the ordering. Given the theory φ, algorithm *directional resolution* process the buckets in a reverse order of d. When processing $bucket_i$, it resolves over Q_i all possible pairs of clauses in the bucket and insert the resolvents into the appropriate lower buckets.

Theorem 1: (model generation)
Let φ be a cnf formula, $d = Q_1, \ldots, Q_n$ an ordering, and $E_d(\varphi)$ its directional extension. Then, if the extension is not empty, any model of φ can be generated in time $O(|E_d(\varphi)|)$ in a backtrack-free manner, consulting $E_d(\varphi)$, as follows: Step 1. Assign to Q_1 a truth value that is consistent with clauses in $bucket_1$ (if the bucket is empty, assign Q_1 an arbitrary value); Step i. After assigning a value to $Q_1, \ldots, Q_{i-1}$, assign to Q_i a value that, together with the previous assignments, will satisfy all the clauses in $bucket_i$. □

Proof: Suppose, to the contrary that during the process of model generation there exists a partial model of truth assignments, $q_1, \ldots, q_i$ for the first $i-1$ symbols that satisfy all the clauses in the buckets of $Q_1, \ldots, Q_{i-1}$, and assume that there is no truth value for Q_i that satisfy all the clauses in the bucket of Q_i. Let α and β be two clauses in the bucket of Q_i that clash. Clearly, α and β contain opposite signs of atom Q_i; in one Q_i appears negatively and in the other positively. Consequently, while being processed by directional-resolution, α and β could have been resolved upon, thus resulting in a resolvent that must appear in earlier buckets. Such a clause, if existed, would not have allowed the partial model $q_1, \ldots, q_i$, thus leading to a contradiction. □

Corollary 1: *[Davis and Putnam, 1960] A theory has a non-empty directional extension iff it is satisfiable.* □

Clearly, the effectiveness of directional resolution both for satisfiability and for subsequent query processing depends on the the size of its output theory $E_d(\varphi)$.

Theorem 2: (complexity)
Given a theory φ and an ordering d of its propositional symbols, the time complexity of algorithm directional resolution is $O(n \cdot |E_d(\varphi)|^2)$, where n is the number of propositional letters in the language.

Proof: There are at most n buckets, each containing no more clauses than the final theory, and resolving pairs of clauses in each bucket is a quadratic operation. □

The bound above, although could be loose, demonstrates the dependence of the algorithm's complexity on the size of its resulting output.

Once $E_d(\varphi)$ is compiled, determining the entailment of a single literal involves checking the bucket of that literal first. If the literal appears there as a unit clause, it is entailed; if not, the negation of that literal should be inserted and the algorithm should be restarted from that bucket. If the empty clause is generated in that process, the literal is entailed. To determine the entailment of an arbitrary clause, each literal of the negated clause must be added to its appropriate bucket and processing restarted from the highest such bucket. This suggests that in knowledge bases, whose queries involve a restricted subset of the alphabet, that subset should be processed last by directional resolution. Namely, the symbols of that subset should appear early in the ordering. In summary,

Theorem 3: *(*entailment*)*
Given a directional extension $E_d(\varphi)$ and a constant c, the entailment of clauses involving only the first c symbols in d is polynomial in the size of $E_d(\varphi)$. □

4 Tractable classes

Consider the following two examples demonstrating the effect of ordering on $E_d(\varphi)$.

Example 1: Let $\varphi_1 = \{(B,A), (C,\neg A), (D,A), (E,\neg A)\}$. For the ordering $d_1 = (E,B,C,D,A)$, all clauses are initially contained in *bucket*(A) (highest in the ordering). All other buckets are empty. Following the application of algorithm directional resolution along d_1, we get (note that processing is in the reverse order of d): *bucket*$(D) = \{(C,D),(D,E)\}$, *bucket*$(C) = \{(B,C)\}$, *bucket*$(B) = \{(B,E)\}$. The directional extension along the ordering $d_2 = (A,B,C,D,E)$ is identical to the input theory, however, and each bucket contains at most one clause.

Example 2: Consider the theory $\varphi_2 = \{(\neg A, B), (A, \neg C), (\neg B, D), (C, D, E)\}$. The directional extensions of φ along the ordering $d_1 = (A,B,C,D,E)$ and $d_2 = (D,E,C,B,A)$ are $E_{d_1}(\varphi) = \varphi$ and $E_{d_2}(\varphi) = \varphi \cup \{(B,\neg C), (\neg C, D), (E,D)\}$, respectively.

In Example 1, A appears in all clauses; hence, it potentially can generate new clauses when resolved upon, unless it is processed last (i.e., put first in the order), as in d_2. This shows that the interactions among clauses play an important role in the effectiveness of the algorithm and may suggest orderings that yield smaller extensions. In Example 2, on the other hand, all atoms have the same type of interaction, each (except E) appearing in two clauses. Nevertheless, D appears positive in both clauses and consequently will not be resolved upon; hence, it can be processed first. Subsequently, B and C appear only negatively in the remaining theory and can, likewise, be processed without generating new clauses. In the following, we will provide a connection between the algorithm's complexity and two parameters: a topological parameter, called *induced width*, and a syntactic parameter, called *diversity*.

Note that directional resolution is tractable for 2-*cnf* theories in all orderings, since 2-*cnf* are closed under resolution (the resolvents are of size 2 or less) and because the overall number of clauses of size 2 is bounded by $O(n^2)$. (In this case, unrestricted resolution is also tractable). Clearly, this algorithm is not the most effective one for satisfiability of 2-*cnfs*. Satisfiability for these theories can be decided in linear time [Even et al., 1976]. However, as noted earlier, *DR* achieves more than satisfiability, it compiles a theory that allows model generation in linear time. We summarize:

Theorem 4: *If φ is a 2-cnf theory, then algorithm directional resolution will produce a directional extension of size $O(n^2)$, in time $O(n^3)$.* □

Corollary 2: *Given a directional extension $E_d(\varphi)$ of a 2-cnf theory φ, the entailment of any clause involving the first c symbols in d is $O(c^3)$.* □

4.1 Induced width

Let $\varphi = \varphi(Q_1, \ldots, Q_n)$ be a *cnf* formula defined over the variables $Q_1, \ldots, Q_n$. The *interaction graph* of φ, denoted $G(\varphi)$, is an undirected graph that contains one node for each propositional variable and an arc connecting any two nodes whose associated variables appear in the same clause. The interaction graph of φ_2 is given in Figure 2a. We can bound the size of all

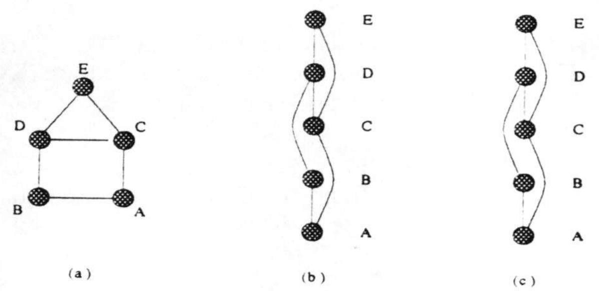


Figure 2: The interaction graph of φ_2

theories having the same interaction graph using some properties of the graph.

Definition 1: Given a graph G and an ordering of its nodes d, the parent set of a node A relative to d is the set of nodes connected to A that precede A in the ordering d. The size of this parent set is the *width* of A relative to d. The width $w(d)$ of an ordering d is the maximum width of nodes along the ordering, and the width w of a graph is the minimal width of all its orderings [Freuder, 1982, Dechter and Pearl, 1987].

Lemma 1: *Given the interaction graph $G(\varphi)$ and an ordering d: If A is an atom having $k-1$ parents, then there are at most 3^k clauses in the bucket of A; if $w(d) = w$, then the size of the corresponding theory is $O(n \cdot 3^w)$.* □

Proof: The bucket of A contains clauses defined on k literals only. For the set of $k-1$ symbols there are at most $\binom{k-1}{i}$ subsets of i symbols. Each subset can be associated with at most 2^i clauses (i.e., each symbol can appear either positive or negative in a clause), and A can be also positive or negative. Therefore we can have at most

$$2 \cdot \sum_{i=0}^{k-1} \binom{k-1}{i} 2^i = 2 \cdot 3^{k-1}. \tag{1}$$

clauses. Clearly, if the parent set is bounded by w, the extension is bounded by $O(n \cdot 3^w)$. □

When applied along d to a theory having graph G, algorithm directional resolution adds clauses and, accordingly, the interaction graph changes.

Definition 2: Given a graph G and an ordering d, the graph generated by recursively connecting the parents of G, in a reverse order of d, is called the *induced graph* of G w.r.t. d and is denoted by $I_d(G)$. The width of $I_d(G)$ is denoted by $w*(d)$ and is called the *induced width* of G w.r.t. d.

The graph in Figure 2a, for example, has width 2 along the ordering A, B, C, D, E (Figure 2b). Its induced graph is given in Figure 2c. The induced width of G equals 2.

Lemma 2: *Let φ be a theory. Then $G(E_d(\varphi))$, the interaction graph of its directional extension along d, is a subgraph of $I_d(G(\varphi))$.*

Proof: The proof is by induction on the symbols along the ordering d. The induction hypothesis is that all the arcs incident to $Q_n, ..., Q_i$ in the $G(E_d(\varphi))$ appear also in $I_d(G(\varphi))$. The claim is true for Q_n, since its connectivity is the same in both graphs. Assume that the claim is true for $Q_n, ..., Q_i$ and we will show that it holds also for Q_{i-1}, namely, if (Q_{i-1}, Q_j) $j < i-1$ is an arc in $G(E_d(\varphi))$, then it is included in $I_d(G(\varphi))$. There are two cases: either Q_{i-1} and Q_j appeared in the same clause of the initial theory, φ, in which case they are connected in $G(\varphi)$ and therefore also in $I_d(G(\varphi))$, or else a clause containing both symbols was introduced during directional resolution. Assume that the clause was introduced while processing bucket $Q_t, t > i-1$. Since Q_{i-1} and Q_j appeared in the bucket of Q_t, each must be connected to Q_t in $G(E_d(\varphi))$ and, by the induction hypothesis, they will also be connected in $I_d(G(\varphi))$. Therefore, Q_{i-1} and Q_j would become connected in $I_d(G(\varphi))$, when connecting the parents of Q_t. □

Theorem 5: *Let $\varphi = \varphi(Q_1, ..., Q_n)$ be a cnf, $G(\varphi)$ its interaction graph, and $w*(d)$ its induced width along d; then, the size of $E_d(\varphi)$ is $O(n \cdot 3^{w*(d)})$.*

Proof: Since the interaction graph of $E_d(\varphi)$ is a subgraph of $I_d(G)$, and since from lemma 1 the size of theories having $I_d(G)$ as their interaction graph is bounded by $O(n \cdot 3^{w*(d)})$, the result follows. Note that this deduction implicitly assumes that the algorithm eliminates duplicate clauses. □

It is known that if a graph is embedded in a k-tree its induced width is bounded by k [Arnborg et al., 1987]. The definition is recursive.

Definition 3: *(k-trees)*
Step 1: A clique of size k is a k-tree.
Step i: given a k-tree defined over $Q_1, ..., Q_{i-1}$, a k-tree over $Q_1, ..., Q_i$ can be generated by selecting a clique of size k and connecting Q_i to every node in that clique.

Corollary 3: *If φ is a formula whose interaction graph can be embedded in a k-tree then there is an ordering d such that the time complexity of directional resolution on that ordering is $O(n \cdot 2^{k+1})$.* □

Finding an ordering yielding the smallest induced width of a graph is NP-hard [Arnborg et al., 1987]. However, any ordering d yields a simple bound, $w*(d)$, of $w*$. Consequently, when given a theory and its interaction graph, we will try to find an ordering that yields the smallest width possible. Several heuristic orderings are available (see [Bertele and Brioshi, 1972]). Important special tractable classes are those having $w* = 1$ (namely, the interaction graph is a tree) and

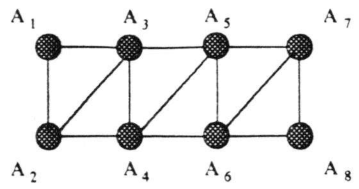


Figure 3: The interaction graph of φ_8 in example 3: $\varphi_8 = \{(A_1, A_2, \neg A_3), (\neg A_2, A_4), (\neg A_2, A_3, \neg A_4), (A_3, A_4, \neg A_5), (\neg A_4, A_6), (\neg A_4, A_5, \neg A_6), (A_5, A_6, \neg A_7), (\neg A_6, A_8), (\neg A_6, A_7, \neg A_8)\}$

those having $w* = 2$, called series parallel networks. These classes can be recognized in linear time. As a matter of fact, given any k, graphs having induced width of k or less can be recognized in $O(exp(k))$.

Example 3: Consider a theory φ_n over the alphabet $\{A_1, A_2, ,..., A_n\}$. The theory φ_n has a set of clauses indexed by i, where a clause for i odd is given by $(A_i, A_{i+1}, \neg A_{i+2})$ and two clauses for i even are given by $(\neg A_i, A_{i+2})$ and $(\neg A_i, A_{i+1}, \neg A_{i+2})$. The reader could check that the induced width of such theories along the natural order is 2 and thus the size of the directional extension will not exceed $18 \cdot n$. For this graph, and for the natural ordering the induced graph is identical to the original graph (see figure 3).

4.2 Diversity

The concept of induced width frequently leads to a loose upper bound on the number of clauses recorded by directional resolution. In example 3 for instance, only 8 clauses were generated by directional-resolution when processed in the natural order, even without eliminating subsumption and tautologies in each bucket, while the computed bound is $18 \cdot 8 = 144$. One source for inaccuracy could be that the induced graph is not a tight bound for the interaction graph of $E_d(\varphi)$. Consider, for instance, the two clauses $(\neg A, B), (\neg C, B)$ and the order $d = A, C, B$. When bucket B is processed, no clause is added because B is positive in both clauses, nevertheless, nodes A and C will be connected in the induced graph. In this subsection, we introduce a more refined parameter, called *diversity*, based on the observation that a propositional letter can be resolved upon only when it appears both positively and negatively in different clauses.

Definition 4: (diversity of a theory)
Given a theory φ and an ordering d, let Q_i^+ (or Q_i^-) denote the number of times Q_i appears positively (or negatively) in $bucket_i$ relative to d. The diversity of Q_i relative to d, $div(Q_i)$, is $Q_i^+ \times Q_i^-$. The *diversity of an ordering d, $div(d)$*, is the maximum diversity of its literals w.r.t. the ordering d and the *diversity of a theory, div*, is the minimal diversity over all its orderings.

```
min-diversity ($\varphi$)
1. For $i = n$ to 1 do
2. Step $i$ (after selecting $Q_{i+1}, ..., Q_n$): choose symbol
   $Q$ having the smallest diversity in $\varphi - \bigcup_{j=i+1}^{n} bucket_j$,
   and put it in the $i$-th position.
3. End.
```

Figure 4: Algorithm *min-diversity*

Theorem 6: *Algorithm* min-diversity *(Figure 4) generates a minimal diversity ordering of a theory.*

Proof: Let d be an ordering generated by the algorithm and let Q_i be a literal whose diversity equals the diversity of the ordering. If Q_i is pushed up, its diversity can only increase and if pushed down, it must be replaced by a literal whose diversity is either equal to or higher than the diversity of Q_i. □

The concept of diversity yields new tractable classes. If d is an ordering having a zero diversity, algorithm directional resolution will add no clauses to φ along d. Namely,

Theorem 7: *Theories having zero diversity are tractable and can be recognized in linear time.* □

Example 4: Let $\varphi = \{(G, E, \neg F), (G, \neg E, D), (\neg A, F), (A, \neg E)\ (\neg B, C, \neg E)\ (B, C, D)\}$. The reader can verify that the ordering $d = A, B, C, D, E, F, G$ is a zero-diversity ordering of φ. Note that the diversity of theories in example 3 along the natural ordering, is 1.

Zero-diversity theories generalize the notion of causal theories defined for general networks of multivalued relations [Dechter and Pearl, 1991]. According to the definition, theories specified in the form of $cnfs$ would correspond to *causal* if there is an ordering of the symbols such that each bucket contains only one clause. Therefore, a causal cnf theory has zero-diversity. Note that even when a general theory is not zero-diversity it is better to put zero-diversity literals last in the ordering (namely they will be processed first). Then, the size of the directional-extension is exponentially bounded in the number of literals having only strictly-positive diversities. In general, however, the parameter of interest is the diversity of the directional extension $E_d(\varphi)$ rather than the diversity of φ.

Definition 5: (induced diversity)
The *induced diversity of an ordering d, $div^*(d)$*, is the diversity of $E_d(\varphi)$ along d, and the *induced diversity of a theory, div**, is the minimal induced diversity over all its orderings.

Since $div*(d)$ bounds the *added* clauses generated from each bucket, we can trivially bound the size of $E_d(\varphi)$ using $div*$: for every d, $|E_d(\varphi)| \le |\varphi| + n \cdot div*(d)$. The problem is that even for a given ordering d, $div*(d)$ is

not polynomially computable, and, moreover, we did not find an effective upper bound. Still it can be used for some special cases. Clearly, for most theories and most orderings
$div*(d) > div(d)$. A special counter example we observed are the zero diversity theories for which $div*(d) = div(d) = 0$. We next identify a subclass of diversity-1 theories whose div* remains 1.

Theorem 8: *A theory $\varphi = \varphi(Q_1, ..., Q_n)$, has $div* \leq 1$ and is therefore tractable, if each symbol Q_i satisfies one of the following conditions: a. it appears only negatively; b. it appears only positively; c. it appears in exactly two clauses.* □

The set of theories in example 3 has $div* = 2$. Note though, that we can easily create examples with high $w*$ having $div* \leq 1$.

4.3 A diversity graph for Horn theories

It is known that general Horn satisfiability can be determined by unit resolution. Note that when DR is processed in a dynamic ordering (as suggested in the original DP-elimination), namely, when propositional letters that appear in a unit clause are processed first (last in the ordering) and when new unit clauses generated, their buckets are pushed up, we have the essence of unit propagation. When incorporating this *dynamic-ordering* variation to directional resolution, satisfiability will be determined polynomially (for Horn theories) if the algorithm terminates once no unit clauses are available. However, executing the algorithm to full completion may result in long output theories [McAllester]. We now show that definite Horn theories of zero diversity can be given a simple graph interpretation, yielding a more accurate estimate of the extension's size for definite and Horn theories.

One may question the usefulness of this exercise since satisfiability is not a problem for Horn theories. Still, directional resolution achieves more than satisfiability, it compiles the Horn theory into a backtrack-free one which might prove useful in some applications, especially those requiring multiple queries on a small subset of the alphabet. For example, in the context of rule-based programs where the rules represent actions to be taken in real time, preprocessing by directional resolution posts constraints that will not allow the execution of rules leading to future deadends. Also, analysis of Horn theories may guide future extensions to general *cnfs* which are near Horn.

Definition 6: *(diversity graph)*
A Horn theory φ can be associated with a *directed* graph called the *diversity graph* and denoted $D(\varphi)$. $D(\varphi)$ contains a node for each propositional letter and an arc is directed from A to B if there is a Horn clause having B in its head (i.e., B is positive) and A in its antecedent (i.e., A is negative). Two special nodes, labeled "true" and "false" are introduced. There is an arc from "true" to A if A is a positive unit clause, and there is an arc from B to "false" if B is included in any negative clause.

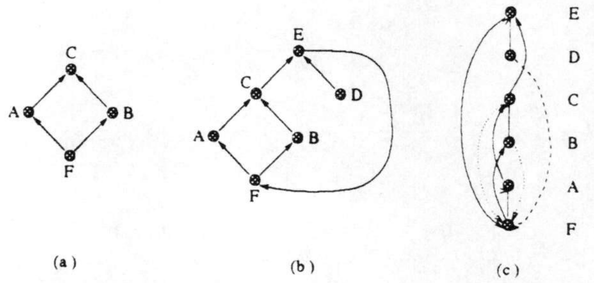


Figure 5: Diversity graphs of Horn theories: a. $D(\varphi_1)$, b. $D(\varphi_2)$, c. the induced diversity graph of φ_2

Example 5: Consider the following two Horn theories: $\varphi_1 = \{A \wedge B \rightarrow C,\ F \rightarrow A,\ \ F \rightarrow B\}$, $\varphi_2 = \{A \wedge B \rightarrow C,\ F \rightarrow A,\ F \rightarrow B,\ C \wedge D \rightarrow E,\ E \rightarrow F\}$. The diversity graphs of φ_1 and φ_2 are presented in Figure 5. We see that φ_1 is an acyclic theory (it has an acyclic diversity graph) while φ_2 is cyclic.

Theorem 9: *A definite Horn theory has an acyclic* diversity graph *iff it has a zero diversity.*

Corollary 4: *If φ is an acyclic definite Horn theory w.r.t. ordering d, then $E_d(\varphi) = \varphi$.* □

Note that the theorem cannot be extended to full Horn theories. For example, the theory

$$\varphi = \{(A \rightarrow B), (\neg A, \neg B), A\}$$

is a Horn theory whose diversity graph is acyclic. Yet it has a non-zero diversity. Note also that definite theories are always satisfiable and they are closed under resolution. We will now show that the notion of a diversity graph will allow a more refined approximation of the directional extension of definite and Horn theories.

Definition 7: *diversity width (div-width)*
Let D be a directed graph and let d be an ordering of the nodes. The positive width of a node Q, denoted $u_+(Q)$, is the number of arcs emanating from prior nodes, called its positive parents, towards Q. The negative width of Q relative to d, denoted $u_-(Q)$, is the number of arcs emanating from Q towards nodes preceding it in the ordering d, called its negative parents. The *diversity-width (div-width)* of Q, $u(Q)$, relative to d is $max\{u_+(Q), u_-(Q)\}$. The *div-width*, $u(d)$, of an ordering, d, is the maximum div-width of each of its nodes along the ordering, and the *div-width* of a Horn theory is the minimum of $u(d)$ over all orderings that starts with nodes " true" and "false".

Lemma 3: *Given a diversity graph of Horn theory $D(\varphi)$, and an ordering d, if A is an atom having k positive parents and j negative parents, then there are at most $O(2^k + j \cdot 2^j)$ non-negative clauses in the bucket of A.* □

A minimum div-width of a graph can be computed by a greedy algorithm like the min-diversity algorithm in figure 4, using div-width criteria for node selection.

As in the case of interaction graph, the diversity graph changes when processed by directional resolution and its diversity graph can be *approximated* by graph manipulation as follows:

Definition 8: *(induced diversity graph and width)* Given a digraph D and an ordering d, such that "true" and "false" appear first, the *induced diversity graph* of D relative to d, denoted $ID_d(D)$, is generated as follows. Nodes are processed from last to first. When processing node Q_i, a directed arc from Q_j to Q_k is added if both nodes precede Q_i in the ordering and if there is a directed arc from Q_j to Q_i and from Q_i to Q_k. The div-width of $ID_d(D)$, denoted by $u*(d)$, is called the *induced diversity width* of D w.r.t. d or *div-width**.

Note that constructing the induced diversity graph is at most $O(n^3)$ when n is the number of vertices.

Example 6: The induced diversity graph of $D(\varphi_2)$ along the ordering $d = F, A, B, C, D, E$ is given in Figure 5. (This is a definite theory, so nodes "true" and "false" are omitted). The added arcs are dotted. The div-width of node E is 2 (its positive div-width is 2 and its negative div-width is 1). In this case, $u(d) = u*(d) = 2$.

We can show:

Lemma 4: *Let φ be a Horn theory and d an ordering of its symbols; then the diversity graph of $E_d(\varphi)$, $D(E_d(\varphi))$, is contained in $ID_d(D(\varphi))$ when d is an ordering which starts with "true" and "false".* □

We can now bound the size of $E_d(\varphi)$ for a Horn theory φ:

Theorem 10: *Let φ be a Horn theory and let d be an ordering of its symbols that starts by "true" and "false", having induced div-width, $u*(d)$ along d; then the size $E_d(\varphi)$ restricted to the non-negative clauses is $O(n \cdot u*(d) \cdot 2^{u*(d)})$ and the size of $E_d(\varphi)$ restricted to the negative clauses is $O(2^{|false|})$, where $|false|$ is the degree of node "false" in the induced diversity graph.*

Proof: Follows immediately from Lemma 3 and Lemma 4. □

Note that the bound on the number of negative clauses may be very loose. Sometimes it will be worse than the bound suggested by the width of the undirected interaction graph. The bound on the number of non-negative clauses though is always more accurate. It is easy to see that for any definite theory, φ, and any ordering d, $w*(d) \geq u*(d)$.

Our earlier observation that *acyclic* diversity graphs of definite theories do not change when processed by directional resolution (using an ordering imposed by the graph), suggests that new arcs are *added* only within *strongly connected components* of the diversity graph. We may, therefore, get a tighter bound on the size of the non-negative clauses added to the directional extension (beyond those in the original theory φ) by consulting each strongly connected component separately.

Definition 9: *(Strongly connected components)* A *strongly connected component* of a directed graph is a maximal set of nodes U such that for every pair A and B in U there is a directed path from A to B and a directed path from B to A. The *component graph* of $G = (V, E)$, denoted $G^{SCC} = (V^C, E^C)$, contains one vertex for each strongly connected component of G, and there is an edge from component A^c to component B^c if there is a directed edge from a node in A^c to a node in B^c in the graph G.

It is well known that the component graph is acyclic and that the strongly connected components can be computed in time linear in the number of vertices and edges of the graph. The connection between the size of the directional extension of a definite theory and its component-based induced div-width is presented in the following theorem. The bound can be extended to Horn theories using the "false" node.

Theorem 11: *Let φ be a definite theory having a diversity graph D. Let $S_1, ..., S_t$ be the strongly connected components of G, let $d_1, d_2, ..., d_t$ be orderings of the nodes in each of the strongly connected components, and let d be a concatenation of the orderings $d = d_{i_1}, ..., d_{i_j}, ..., d_{i_t}$ that agrees with the partial acyclic ordering of the components' graph. Let $u*(d_j)$ be the largest induced div-width of any component. Then, the size of $E_d(\varphi) - \varphi$ is $O(n2^{u*(d_j)})$.* □

Consequently, we can restrict ourselves to *admissible* orderings only: those that agree with the acyclic structure of the component graph. Hence, we can modify the definition of induced div-width of a digraph along such orderings to coincide with the largest induced div-width among its strongly connected components.

Example 7: Consider again the theory φ_1 in Example 5. Since the graph is acyclic, the strongly connected components contain only one node, and therefore for any admissible ordering d, $u*(d) = 0$. Indeed no clause will be added. For theory φ_2 there are

```
DP-backtracking($\varphi$)
Input: A cnf theory $\varphi$.
Output: A decision of whether $\varphi$ is satisfiable.
1. Unit_propagate($\varphi$);
2. If the empty clause generated return(false);
3. else if all variables are assigned return(true);
4. else
5.      $Q$ = some unassigned variable;
6.      return( DP-backtracking( $\varphi \wedge Q$) $\vee$
7.                DP-backtracking($\varphi \wedge \neg Q$) )
```

Figure 6: DP-backtracking algorithm

two components, one including D only and another including the rest of the variables. For the ordering $d = F, A, B, C, E$ on that component, only the arcs $(C, F), (B, F)(A, F)$ will be added, resulting in an induced div-width of 2 (see Figure 5c).

To conclude, the main purpose of the analysis in this section is to determine ahead of time the usefulness of algorithm directional resolution for a given theory and, more importantly, to suggest a good heuristic ordering that may result in a small induced width, small diversity, or small induced div-width for Horn theories. We know that finding an optimal width is NP-hard, and we conjecture that finding an optimal induced div-width is also hard, nevertheless good orderings can be generated using various heuristics (like min-width, min-diversity and min-div-width).

5 Bounded directional resolution

Since algorithm directional resolution is time and space exponential in the worst case, we propose an approximate algorithm called *bounded directional resolution* (BDR). The algorithm records clauses of size k or less when k is a constant. Consequently, its complexity is polynomial in k. Algorithm bounded directional resolution parallels algorithms for directional k-consistency in constraint satisfaction problems [Dechter and Pearl, 1987].

6 Experimental evaluation

DP-backtracking has been implemented in C language as a variant of the Davis-Putnam procedure (see Figure 6).

It has been augmented with the *2-literal clause heuristic* proposed in [Crawford and Auton, 1993] which prefers a variable that would cause the largest number of unit propagations. The number of possible unit propagations is approximated by the number of 2-literal clauses in which the variables appear. The modified version significantly outperforms DP-backtracking without this heuristic [Crawford and Auton, 1993]. In order to find a solution following DR we ran DP-backtracking using the reverse ordering of variables used by DR, but without the 2-literal clause heuristic. The reason is that we wanted to fix the order of variables. As theory dictates, no deadends occur when DP-backtracking is applied after DR on the same ordering. In this case DP-backtracking takes linear time in the extension size.

Algorithm BDR, since it is incomplete for satisfiability, was followed by DP-backtracking augmented with the 2-literal clause heuristic.

Different orderings of variables were used by the algorithms: input ordering as used by the generator of problems, min-width ordering and min-diversity ordering. Given an interaction graph, min-width ordering selects a variable with the smallest degree, and puts it last in the ordering; the node is eliminated from the graph and the ordering continues recursively. Min-diversity ordering have been described above.

In conjunction with DR we have experimented with both static and dynamic orderings. Static orderings were computed prior to search while dynamic orderings were computed at each step of the search. We report the results on static orderings only since we did not observe any significant difference in DR's efficiency when running the algorithms on both dynamic and static orderings.

Several random generators were used in order to test the algorithms over problems with different structure. To generate *uniform k-cnfs* we used the generator proposed by [Mitchell et al., 1992] taking as input the number of variables n, the number of clauses m, and the number of literals per clause k. We generate each clause randomly choosing k variables from the set of n variables and by determining the polarity of each literal with probability 0.5. Our second generator, called *mixed cnf generator*, generates theories containing clauses of length k_1 and clauses of length k_2. The third generator, called *chains*, first used the *uniform k-cnf* generator to obtain a sequence of n independent random subtheories, and then connected all the subtheories in a chain by generating 2-cnf clauses using one variable from the i-th subtheory and one from the $(i + 1)$-th subtheory. Similarly we also connected the n independent subtheories into a tree structure. The obtained results were similar to those on chains, so we report only the result on chains. We experimented also with random *embeddings in k-trees* [Arnborg et al., 1987]. However, we were unable to generate hard instances with more than few deadends. Consequently, the performance of both DR and DP-backtracking was similarly efficient.

We measured CPU time for all algorithms, and the number of deadends for DP-backtracking as characteristics of problems' difficulty. We measured also the number of new clauses generated by DR, the maximal

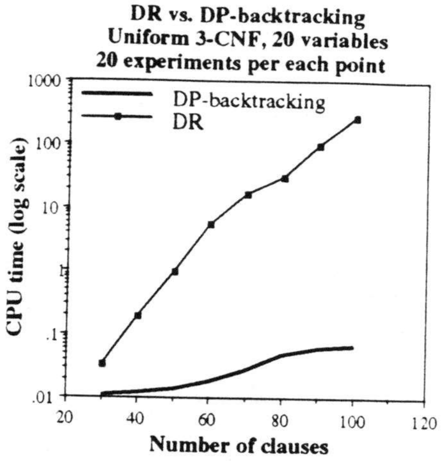


Figure 7: DR and DP-backtracking on 3-$cnfs$

size of generated clauses, and the induced width. The number of experiments shown in the figures is usually per each point unless stated otherwise.

6.1 Results for Problems with Uniform Structure

We compared DP-backtracking with DR on randomly generated *k-cnfs* for k=3,4,5 and on mixed theories. In all these cases DP-backtracking significantly outperforms DR. It is observed that the complexity of DR indeed grows exponentially with the size of problems (see Figure 7). We show the results for 3-$cnfs$ with 20 variables only. On larger problems DR often ran out of memory because of the large number of generated clauses.

Since DR was so inefficient for solving uniform k-$cnfs$ we next experimented with Bounded Directional Resolution (BDR) using different bounds. Our experiments show that when the input theory is a uniform k-cnf and BDR uses a bound less than k, almost no new clauses are added. On the other hand, when the bound is strictly greater than k, the preprocessing phase of BDR by itself is considerably worse than DP-backtracking. The only promising case occurs when the bound equals k. We observed that in this case relatively few clauses were added by BDR which therefor ran much faster. Also, DP-backtracking often ran a little bit faster on the generated theory and therefore the combined algorithm was slightly more efficient than DP-backtracking alone (see Figure 8).

6.2 Results for Chains

The behaviour of the algorithms on chains differs dramatically from that on uniform instances. We found extremely hard instances for DP-backtracking, orders of magnitude harder than those generated by the uniform model. In the Table 1 we compare performance

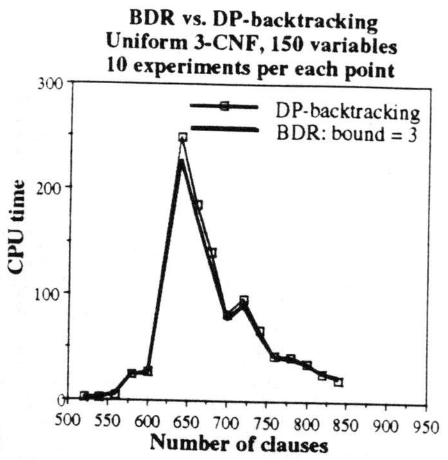


Figure 8: BDR with bound=3 and DP-backtracking on 3-$cnfs$

of DP-backtracking on uniform 3-*cnf* problems and on 3-*cnf* chain problems of the same size. Chain problems contain 25 subtheories with 5 variables and 9 to 23 3-*cnf* clauses per subtheory, together with 24 2-*cnf* clauses connecting subtheories in the chain. The corresponding uniform 3-*cnf* problems have 125 variables and 249 to 599 clauses. We tested DP-backtracking on both classes of problems. The table shows mean values on 20 experiments where the number of experiments is per a constant problem size. We used min-diversity ordering for each instance.

First, we observed extremely hard chain problems with many deadends around the cross-over point for chains, orders of magnitude harder than uniform 3-cnf problems of the same size. Second, we note that the crossover point for chain problems is shifted towards a smaller number of clauses per number of variables.

Table 1: DP on uniform 3-$cnfs$ and on chain problems of the same size: 125 variables

Mean values on 20 experiments						
Num of clauses	**Uniform 3-cnfs**			**3-cnf chains**		
	% Sat	Time 1st solution	Dead ends	% Sat	Time 1st solution	Dead ends
249	100	0.2	0	100	0.3	0
299	100	0.2	0	100	0.4	1
349	100	0.2	3	70	9945.7	908861
399	100	0.2	2	25	2551.1	207896
449	100	0.4	17	15	185.2	13248
499	95	3.7	244	0	2.4	160
549	35	8.5	535	0	0.9	9
599	0	6.6	382	0	0.1	6

On the other hand, DR behaved in a tamed way on the chain problems and was sometimes more than 1000

Table 2: DR and DP on 3-*cnf* chains: 25 subtheories, 5 variables in each

Min-diversity ordering (static)											
Mean values on 20 experiments											
Num of variables	Num of clauses	% Sat problems	Time: 1st solution, DP-backtracking	Number of dead ends	Time: SAT only, DR	Time: 1st solution, DP after DR	Dead ends after DR	Time: SAT+1st solution, DR	Number of new clauses	Size of Max clause	Iduced width
125	249	100	0.34	0.2	0.64	0.30	0.0	1.10	61.4	4.1	5.1
125	299	100	0.41	1.4	1.42	0.32	0.0	1.92	105.2	4.1	5.3
125	349	70	9945.69	908861.2	2.23	0.33	0.0	2.72	130.8	4.0	5.3
125	399	25	2551.09	207896.3	2.79	0.19	0.0	3.08	131.1	4.0	5.3
125	449	15	185.19	13248.1	3.67	0.27	0.0	4.12	135.4	4.0	5.5
125	499	0	2.43	159.6	3.84	0.00	0.0	3.84	116.2	3.9	5.4
125	549	0	0.18	9.4	4.03	0.00	0.0	4.03	99.0	3.9	5.2
125	599	0	0.14	6.1	4.59	0.00	0.0	4.59	93.2	3.6	5.2

times faster than DP-backtracking. In Table 2 we compare DP-backtracking with DR on the same chain problems as in Table 1 for finding one solution and for deciding satisfiability only. A more detailed illustration in Table 3 lists the results on selected hard instances from Table 2 (number of deadend exceeds 4000).

Table 3: DR and DP on hard instances (number of deadends > 4000): 3-*cnf* chains with 125 variables

Num of cls	SAT: 0 or 1	**DP-backtracking** Time: 1st solution	Dead ends	**DR** Time: 1st solution
349	0	41163.8	3779913	1.5
349	0	102615.3	9285160	2.4
349	0	55058.5	5105541	1.9
349	0	21.2	2050	2.4
399	0	74.8	6053	3.6
399	0	87.7	7433	3.1
399	0	149.3	12301	3.1
399	0	37903.3	3079997	3.0
399	0	11877.6	975170	2.2
399	0	52.0	4215	3.3
399	0	841.8	70057	2.9
449	1	655.5	47113	5.2
449	0	60.5	4359	4.7
449	0	2549.2	181504	3.0
449	0	289.7	21246	3.5

As expected, DR significantly outperforms DP-backtracking for instances in which DP-backtracking encountered many deadends. Figure 9a shows that the CPU time of DP-backtracking grows linearly with the numbers of deadends (note, that we use logarithmic scale for CPU time) while in case of DR it remains almost constant. We have displayed CPU time on problem instances hard for DP-backtracking (the number of deadends is greater than 1000).

All the experiments before used min-diversity ordering. When experimenting with different orderings (input and min-width) we observed similar results (Figure 9b,c).

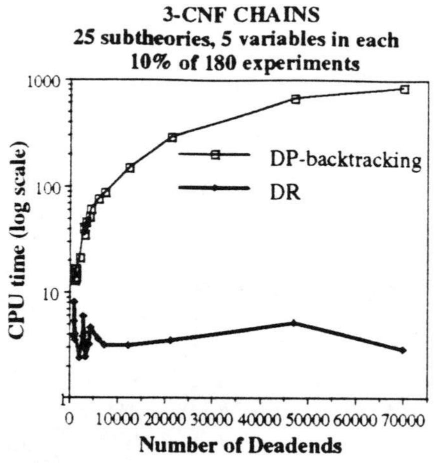
We also experimented a little with the actual code of ***tableau*** [Crawford and Auton, 1993], Crawford and Auton's implementation of Davis-Putnam procedure with various heuristics. We observed a similar behaviour on chain problems. Although some problem instances hard for our version of DP-backtracking were easy for tableau, others were extremely difficult for both algorithms.

We see that almost all the hard chain problems for DP-backtracking were unsatisfiable. Here is a possible explanation. Suppose there is an unsatisfiable subtheory U in a chain problem whose variables are put at the end of an ordering. If all the other subtheories are satisfiable, then DP-backtracking will try to re-instatiate variables from the satisfiable subtheories each time it encounters a deadend. Not knowing the structure hurts DP-backtracking.

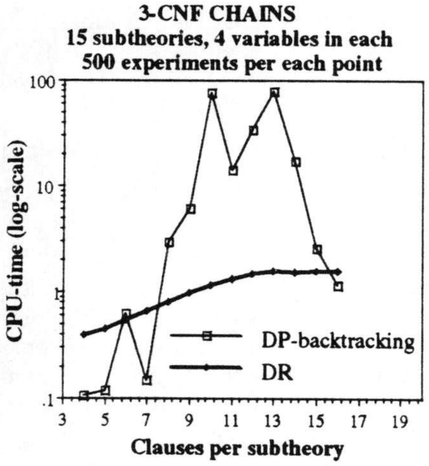
Choosing the right ordering would help but this may be hard to recognize without some preprocessing. Other variants of backtracking that are capable of exploiting the structure like ***backjumping*** [Dechter, 1990] would avoid useless re-instantiation of variables sometimes performed by DP-backtracking . Experiments with backjumping on the same chain instances as used in Table 2 showed that all the problems that were hard for DP-backtracking were quite easy for backjumping (see Figure 10). Backjumping also outperforms DR.

7 Related work and conclusions

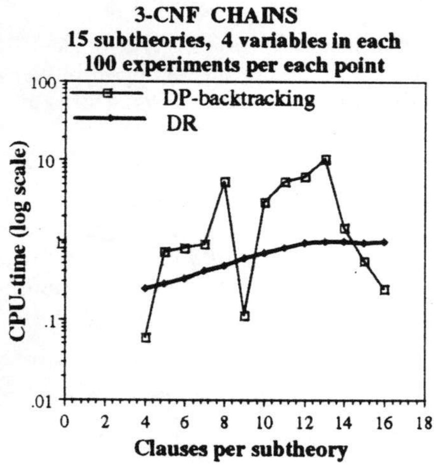
Directional resolution belongs to a family of elimination algorithms first analyzed for optimization tasks in dynamic programming [Bertele and Brioshi, 1972] and later used in constraint satisfaction [Seidel, 1981. Dechter and Pearl, 1987] and in belief networks [Lauritzen and Spigelholter, 1988]. The complexity of all these elimination algorithms can be bounded as a function of the induced width $w*$ of the undirected graph characteristic of each problem instance. Although it is known that determining the $w*$ of an arbitrary graph is NP-hard, useful heuristics for bounding $w*$ are available.



(a) hard instances: more than 1000 deadends



(b) input ordering



(c) min-width ordering

Figure 9: DR and DP-Backtracking on chains

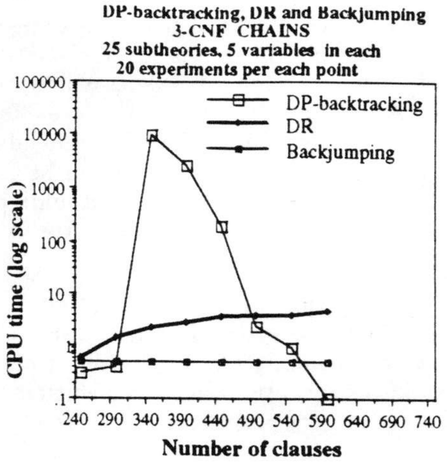


Figure 10: DP-Backtracking, DR and Backjumping on chains: static min-diversity ordering

Since propositional satisfiability is a special case of constraint satisfaction, the induced-width bound could be obtainedby mapping a propositional formula into the relational framework of a constraint satisfaction problem (see [Ben-Eliyahu and Dechter, 1991]), and applying and applying *adaptive consistency*, the elimination algorithm tailored for constraint satisfaction problems [Dechter and Pearl, 1987, Seidel, 1981]. We have recently shown, however, that this kind of pair wise elimination operation as performed by directional resolution is more effective. And, while it can be extended to any row-convex constraint problem [Van Beek and Dechter, 1993] or to every 1-tight relations [Van Beek and Dechter, 1993] it cannot decide consistency for arbitrary multi-valued networks of relations.

Specifically the paper makes three main contributions. First, we revive the old Davis-Putnam algorithm (herein called *directional resolution*). Second, we mitigate the pessimistic analyses of DP-elimination by showing that algorithm *directional resolution* admits some known tractable classes for satisfiability and constraint satisfaction, including 2-*cnfs*, Horn clauses, causal networks, and bounded-width networks. In addition, we identify new tractable classes based on the notion of *diversity* , and show a tighter bound for the size of the directional extension of Horn theories based on *induced diversity width*. Finally, Our empirical tests show that, while on uniform theories directional resolution is ineffective, on problems with special structures, like *chains*, namely with low $w*$, directional resolution greatly outperforms DP-backtracking which is one of the most effective satisfiability algorithm known to date.

In conclusion, although directional resolution outperformed DP-backtracking on some classes of problems, it is not advocated as an effective method for general satisfiability problems. Even when the structure is right, there are other structure-exploiting algorithms, like backjumping, that may be more effective in finding a satisfying solution. What we do advocate is that structure-based components should be integrated, together with other heuristics (like unit propagation), into any algorithm that tries to solve satisfiability effectively.

At the same time, we have shown that, for some structured domains, directional resolution is an effective knowledge compilation procedure. It compiles knowledge into a form that facilitates efficient model generation and query processing.

Acknowledgements

We would like to thank Dan Frost for experimenting with backjumping, Eddie Schwalb and Rachel Ben-Eliyahu for comments on this paper.

References

[Arnborg et al., 1987] S. Arnborg, D.G. Corneil and A. Proskurowski, "Complexity of finding embedding in a k-tree", *SIAM Journal of Algebraic Discrete Methods*, 8(2), 1987, pp. 177-184.

[Bertele and Brioshi, 1972] U. Bertele and F. Brioschi, *Nonserial Dynamic Programming*, Academic Press, New York, 1972.

[Ben-Eliyahu and Dechter, 1991] R. Ben-Eliyahu and R. Dechter, "Default logic, propositional logic and constraints", in *Proceedings of the National Conference on Artificial Intelligence* (AAAI-91), July 1991, Anaheim, CA, pp. 379-385.

[Crawford and Auton, 1993] J. Crawford and L. Auton, "Experimental results on the crossover point in satisfiability problems", in *Proceedings of AAAI-93*, 1993, pp 21-27.

[Davis et al., 1962] M. Davis, G. Logemann and D. Loveland, "A machine program for theorem proving", *Communications of the ACM*, 5, 1962, pp. 394-397.

[Davis and Putnam, 1960] M. Davis and H. Putnam, "A computing procedure for quantification theory", *Journal of the ACM*, 7, 1960, pp. 201-215.

[Dechter and Pearl, 1987] R. Dechter and J. Pearl, "Network-based heuristics for constraint satisfaction problems", in *Artificial Intelligence*, 34, 1987, pp. 1-38.

[Dechter and Pearl, 1991] R. Dechter and J. Pearl, "Directed constraint networks: A relational framework for causal models", in *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence* (IJCAI-91), Sidney, Australia, August 1991, pp. 1164-1170.

[Dechter, 1990] R. Dechter, "Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition", *Artificial Intelligence*, 41, 1990, 273-312.

[Even et al., 1976] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow", *SIAM Journal on Computing*, 5, 1976, 691-703.

[Freuder, 1982] E.C. Freuder, "A sufficient condition for backtrack-free search", *Journal of the ACM*, 29, 1982, 24-32.

[Galil, 1977] Z. Galil, "On the complexity of regular resolution and the Davis-Putnam procedure", *Theoretical Computer Science* 4, 1977, 23-46.

[Goerdt, 1992] A. Goerdt, "Davis-Putnam resolution versus unrestricted resolution", *Annals of Mathematics and Artificial Intelligence*, 6, 1992, 169-184.

[Goldberg et al., 1982] A. Goldberg, P. Purdom and C. Brown, "Average time analysis of simplified Davis-Putnam procedures", *Information Processing Letters*, 15, 1982, 72-75.

[McAllester] D. McAllester, Private communication

[Mitchell et al., 1992] D. Mitchell, B. Selman and H. Levesque, "Hard and Easy Distributions of SAT Problems", in *Proceedings of AAAI-92*, 1992.

[Seidel, 1981] R. Seidel, "A new method for solving constraint satisfaction problems", in *Proceedings of the Seventh international joint conference on Artificial Intelligence* (IJCAI-81), Vancouver, Canada, August 1981, pp. 338-342.

[Selman, 1992] B. Selman, H. Levesque and D. Mitchell, "A new method for solving hard satisfiability problems", in *Proceedings of the Tenth National Conference on Artificial Intelligence* (AAAI-92), San Jose, CA, July 1992.

[Lauritzen and Spigelholter, 1988] S.L. Lauritzen and D.J. Spigelholter, "Local computations with probabilities on graphical structures and their applications to expert systems", *Journal of the Royal Statistical Society, Series, B,* 50, 1988, pp. 65-74.

[Van Beek and Dechter, 1993]
P. van Beek and R. Dechter. On the minimality and decomposability of row-convex constraint networks, June, 1993. Submitted manuscript.

[Van Beek and Dechter, 1993]
P. van Beek and R. Dechter. Constraint tightness vs global consistency November, 1993. Submitted manuscript.