2-way *vs.* d-way Branching for CSP

Joey Hwang and David G. Mitchell

School of Computing Science, Simon Fraser University
Burnaby, V5A 1S6, Canada
{jhwang,mitchell}@cs.sfu.ca

Abstract. Most CSP algorithms are based on refinements and extensions of backtracking, and employ one of two simple "branching schemes": 2-way branching or d-way branching, for domain size d. The schemes are not equivalent, but little is known about their relative power. Here we compare them in terms of how efficiently they can refute an unsatisfiable instance with optimal branching choices, by studying two variants of the resolution proof system, denoted *C-RES* and *NG-RES*, which model the reasoning of CSP algorithms. The tree-like restrictions, *tree-C-RES* and *tree-NG-RES*, exactly capture the power of backtracking with 2-way branching and d-way branching, respectively. We give a family instances which require exponential sized search trees for backtracking with d-way branching, but have size $O(d^2n)$ search trees for backtracking with 2-way branching. We also give a natural branching strategy with which backtracking with 2-way branching finds refutations of these instances in time $O(d^2n^2)$. The unrestricted variants of *C-RES* and *NG-RES* can simulate the reasoning of algorithms which incorporate learning and k-consistency enforcement. We show exponential separations between *C-RES* and *NG-RES*, as well as between the tree-like and unrestricted versions of each system. All separations given are nearly optimal.

1 Introduction

Most complete algorithms for solving finite-domain constraint satisfaction problems (CSPs) are based on backtracking, usually refined with various propagation schemes and sometimes extended with no-good learning. These algorithms are based on one of two "branching schemes". Most CSP papers study algorithms using d-way branching, in which an instance $\mathcal{I}$ is solved as follows. Select a variable x with domain $D(x) = \{1, 2, \ldots, d\}$. For each $a \in D(x)$, we restrict $\mathcal{I}$ by setting $x = a$, and recursively try to solve the restricted instance. $\mathcal{I}$ has no solution if and only if none of the d restricted versions do. In 2-way branching, we select variable x and a value $a \in D(x)$, and make two recursive calls. The first is with the restriction $x = a$; the second with the value a removed from the domain of x. $\mathcal{I}$ has no solution if neither recursive call finds a solution.

It is easy to check that any strategy for d-way branching can be simulated by a 2-way branching strategy with no significant loss of efficiency. But does the converse hold, or is d-way branching strictly more powerful than 2-way branching? Many practitioners believe that 2-way branching is more powerful, and several

commercial solvers support use of this scheme, but little is known about how much more power might be available. It was shown in [11] that 2-way branching with learning is strictly more powerful than d-way branching with learning. In particular, a family of instances MPH_n was given, having the following property: Any d-way branching algorithm, even with optimal variable ordering and optimal learning strategy, cannot solve MPH_n in less than $n^{\Omega(\log n)}$ time, but there is a 2-way branching algorithm, with fairly simple branching and learning strategies, that solves MPH_n in time $O(n^3)$. This leaves open the question of the relative power of the branching schemes without learning, and also the question of whether a true exponential separation can be obtained in the case with learning.

Here, we answer the first question by giving instances which require exponential size search trees for backtracking with d-way branching, but are solved in polynomial time by backtracking with 2-way branching. We take a significant step toward answering the second, by giving instances establish an exponential separation between two proof systems which can simulate the algorithms of interest. However, it is an open question whether the classes of algorithms in question are as powerful as the proof systems. The analogous question in the context of SAT is whether or not conflict-directed clause learning is as powerful as unrestricted resolution, and is also open [2]. We also do not give an efficient strategy for finding the short 2-way branching proofs, although we believe there is one. Figure 1 summarizes our results on the proof systems.

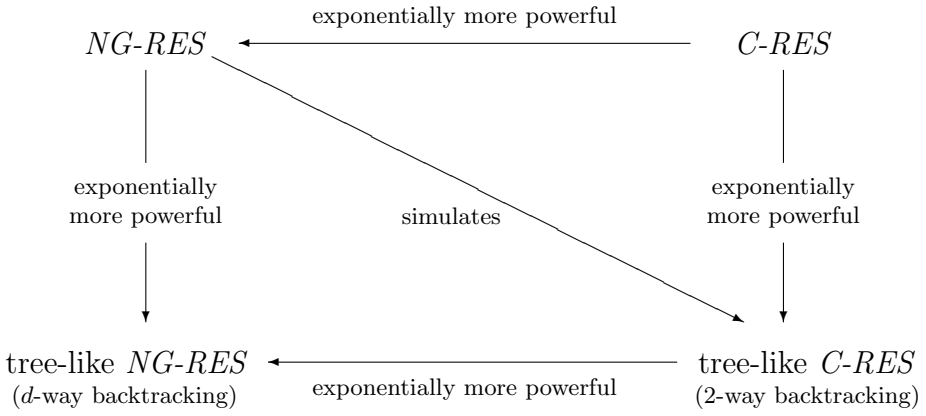

Fig. 1. Relative Efficiency of *NG-RES*, *C-RES* and their tree-like variants

A few experimental empirical studies on 2-way and d-way branching strategies have been reported. Park [12] showed that in most cases, with "standard" variable and value ordering heuristics, 2-way branching ends up simulating d-way branching. To see why, consider pure 2-way backtracking with branching based on "smallest domain first". Once a variable x is branched on, the following branches will also be on x. This simple reasoning does not directly generalize

to more interesting cases, but does give some intuition. Smith and Sturdy [13] investigated the effect of changing the value ordering in 2-way branching, comparing performance to d-way branching. Their finding was that 2-way branching, even with the worst value ordering, is not worse than d-way branching. These studies, combined with our results, suggest that designers of heuristics should consider the properties needed to take advantage of the extra power available with 2-way branching.

Formally, we study only unsatisfiable instances, because an optimal strategy will solve satisfiable instances trivially. However, it would be wrong to think the results say nothing about satisfiable instances. Any polytime branching strategy will make bad choices, after which unsatisfiability of a restricted instance must be proven. Indeed, a reasonable backtracking algorithm can take a non-trivial amount of time only in this way.

2 Preliminaries

Constraint Satisfaction Problems A CSP instance $\mathcal{I}$ is a triple $\langle X, \mathcal{D}, \Gamma \rangle$ where X is a finite set of variables, $\mathcal{D}(x)$ is the domain of a variable $x \in X$, and Γ is a set of nogoods. A *literal* is an expression of the form $x = a$, where $x \in X$ and $a \in D(x)$, and a *nogood* is a set of literals with distinct variables. We write nogoods as $\eta(x_1 = a_1, x_2 = a_2, \cdots, x_t = a_t)$. A (partial) assignment α for $\mathcal{I}$ is a (partial) function from variables to domain values. Assignment α *satisfies* a nogood N iff for some literal $(x = a) \in N$, $\alpha(x)$ is defined and $\alpha(x) \neq a$. A (total) assignment α satisfies $\mathcal{I}$ if there is no nogood $N \in \Gamma$ such that for each $(x = a) \in N$, $\alpha(x) = a$. We denote the set of variables occurring in a nogood N, set of nogoods Γ, or instance $\mathcal{I}$, by $vars(N)$, $vars(\Gamma)$ and $vars(\mathcal{I})$, respectively. We usually assume that all variables of an instance have the same domain, which is the set $[d] = \{1, \ldots d\}$.

Our choice to describe the constraints as a set of nogoods, rather than the usual scope–relation pairs, is purely for convenience. The size of this set as an encoding of a constraint is not relevant to our results, and we could as well assume that the encoding of each constraint is of zero size, with nogoods generated explicitly only as needed.

Propositional Resolution (*RES*) The resolution rule allows us to derive the clause $A \vee B$ from two clauses $x \vee A$ and $\overline{x} \vee B$. We will usually write clauses parenthesized and with $\vee$'s omitted, for example writing $(a\ b\ c)$ rather than $a \vee b \vee c$. A *resolution derivation* of clause C from CNF formula ϕ is a sequence of clauses $C_1, C_2, \cdots, C_m$ in which each C_i is either a clause of ϕ or is derived by the resolution rule from some C_j and C_k with $j, k < i$, and $C_m = C$. A resolution derivation of the empty clause from ϕ is called a *resolution refutation* of ϕ. A CNF formula ϕ is unsatisfiable iff it has a resolution refutation.

Nogood Resolution (*NG-RES*) If the domain of variable x is $\{1, 2, \cdots, d\}$, the *nogood resolution rule* allows one to infer a nogood from a set of nogoods by

resolving on x:

$$\frac{\begin{array}{c}\eta(x=1,\ N_1)\\ \eta(x=2,\ N_2)\\ \vdots\\ \eta(x=d,\ N_d)\end{array}}{\eta(N_1, N_2, \ldots N_d)}\ x \in \{1, \ldots, d\}$$

A *nogood resolution derivation* of a nogood N from a CSP instance Γ is a sequence of nogoods $N_1, N_2, \cdots, N_m$ in which each nogood N_i is either in Γ or is derived from a set of previous nogoods in the sequence by the nogood resolution rule, and $N_m = N$. A *nogood resolution refutation* of $\mathcal{I}$ is a nogood resolution derivation of the empty nogood, $\square$, from Γ. *NG-RES* is a sound and complete refutation system, first proposed in [1].

Constraint Resolution (*C-RES*) Let $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ be a CSP instance. We encode $\mathcal{I}$ as a CNF formula, CNF($\mathcal{I}$), as follows. For each variable $x \in \mathit{vars}(\mathcal{I})$ and each value $a \in \mathcal{D}(x)$, we introduce a propositional variable $x\!:\!a$ asserting that x takes value a when $x\!:\!a$ is true. We have a set of *domain clauses*, ensuring that every variable in $\mathcal{I}$ is given a value, and a set of *unique value clauses* ensuring no variable takes multiple values. For each nogood $N \in \Gamma$, CNF($\mathcal{I}$) has a *constraint clause* which rules assignment forbidden by N. The CNF encoding of $\mathcal{I}$ is:

$$\text{CNF}(\mathcal{I}) = \mathit{domainCls} \cup \mathit{uniqueValueCls} \cup \mathit{constraintCls}$$

$$\begin{aligned}\text{where}\quad \mathit{domainCls} &= \{(x\!:\!a_1\ \cdots\ x\!:\!a_d) : x \in \mathit{vars}(\mathcal{I}), \mathcal{D}(x) = \{a_1, \cdots, a_d\}\}\\ \mathit{uniqueValueCls} &= \{(\overline{x\!:\!a}\ \ \overline{x\!:\!c}) : x \in \mathit{vars}(\mathcal{I}), a, c \in \mathcal{D}(x), a \neq c\}\\ \mathit{constraintCls} &= \{(\overline{x_1\!:\!a_1}\ \cdots\ \overline{x_k\!:\!a_k}) : \eta(x_1 = a_1, \cdots, x_k = a_k) \in \Gamma\}.\end{aligned}$$

There is a one-to-one correspondence between solutions of $\mathcal{I}$ and satisfying truth assignments for CNF($\mathcal{I}$). A *constraint resolution* (*C-RES*) refutation of a CSP instance $\mathcal{I}$ is a *RES* refutation of CNF($\mathcal{I}$), and clearly $\mathcal{I}$ is unsatisfiable iff it has a *C-RES* refutation.

Derivations, Graphs and Tree-like Proofs For T one of the refutation systems defined above, and π a derivation in T (T-derivation), we define the *graph of* π to be the directed acyclic graph (DAG) G_π in which vertices are nogoods (or clauses, as appropriate) of π and there is an edge from vertex v to vertex u if v is a premise for deriving u in π. Derivation π is *tree-like* if every vertex in G_π has out-degree 0 or 1, or equivalently, every derived nogood or clause is used at most once as a premise to derive another. We denote the restriction of T to tree-like derivations by *tree-*T. For example, a *tree-C-RES* refutation of $\mathcal{I}$ is a tree-like resolution refutation of CNF($\mathcal{I}$).

Proof Complexity Let T be one of the proof systems defined above, and π be a T-derivation. The *size* of π, $|\pi|$, is the number of clauses or nogoods in π, as appropriate to T. The *width* of a clause or nogood C, $w(C)$, is the number of literals appearing in C and the width of a derivation π, $w(\pi)$, is the width of the widest clause or nogood in π. The T-complexity of formula or CSP instance ϕ, denoted $T(\phi)$, is the size of its smallest T-refutation. *tree-*$T(\phi)$ is the size of the smallest tree-like refutation.

We say that a proof system A *p-simulates* a proof system B if there is a function that maps any B refutation of a CSP instance $\mathcal{I}$ to some A refutation of $\mathcal{I}$ with at most polynomial blowup in size. We say that A *efficiently simulates* B if the degree of the polynomial in the p-simulation is small. There is an *exponential separation* of system B from system A if there is an infinite set of instances $\{\mathcal{I}_1, \mathcal{I}_2, \cdots\}$ such that the smallest B refutation of $\mathcal{I}_n$ is of size exponential in n, but the smallest A refutation of $\mathcal{I}_n$ is of size polynomial in n. If A efficiently simulates B and there is an exponential separation of B from A, then A is exponentially more powerful than B. For example, it is known that unrestricted resolution is exponentially more powerful than tree-like resolution [4, 3].

Refutations and Algorithms It is straightforward to show that the size of minimal refutations in *tree-RES*, *tree-NG-RES*, and *tree-C-RES* are the same as the size of minimal search trees for backtracking for SAT (DPLL), CSP backtracking with d-way branching, and CSP backtracking with 2-way branching, respectively. This remains true when techniques such as unit propagation, forward checking, and conflict-directed backjumping are employed [1, 10, 11].

3 Results

3.1 Separating Instances for *tree-NG-RES*

Our first result is an exponential separation between *tree-NG-RES* and *NG-RES*. This involves exhibiting instances which require large *tree-NG-RES* refutations, but have short *NG-RES* refutations. The same instances we use for this also have short *tree-C-RES* refutations, and thus also provide the exponential separation between *tree-NG-RES* and *tree-C-RES*. The instances are based on directed acyclic graphs, generalizing the implication graph formulas used in [3] to provide a near-optimal separation of *RES* and *tree-RES*.

Definition 1 (Implication Graph Contradictions ($IMP_{G,S,T,d}$)). *Call a DAG in which every vertex has in-degree 2 or 0 a* circuit. *Let $G = (V, E)$ be a circuit with vertices $\{v_1, \ldots, v_n\}$, $d \geq 3$ be an integer, S the set of sources of G, and T the sets targets (sinks) of G. We associate to each vertex $v_i \in V$ a variable x_i. The* implication graph contradiction *of G, $IMP_{G,S,T,d}$, is the CSP instance with variables $x_1, \cdots, x_n$, having domain $[d]$, and the following nogoods:*

Source axioms: $\eta(x_i = 1)$ *for every $v_i \in S$*
Target axioms: $\eta(x_i = a)$ *for every $v_i \in T$, and for all $a \in [d] \setminus \{1\}$*

Pebbling axioms: $\eta(x_i = a,\ x_j = b,\ x_k = 1)$ *for every* v_k *with predecessors* v_i *and* v_j*, and for all* $a, b \in [d] \setminus \{1\}$

$IMP_{G,S,T,d}$ expresses the following contradiction: Each vertex of G is labeled with a number in $[d]$. Sources are not labeled 1, and if neither predecessor of an internal vertex v is labeled 1 then v is not labeled 1, but targets are labeled 1.

Theorem 1. *There exists an infinite family of* n*-vertex circuits* G*, with sources* S *and targets* T*, such that for any integer* $d \geq 3$*, tree-NG-RES*$(IMP_{G,S,T,d}) = (d-1)^{\Omega(n/\log n)}$.

The proof is given in Section 4.

Theorem 2. *For any* n*-vertex circuit* G*, with sources* S *and targets* T*, and any integer* $d \geq 3$*, NG-RES*$(IMP_{G,S,T,d}) = O(d^2 n)$.

The following algorithm constructs such refutations.

Efficient 2-way branching for $IMP_{G_n,S,T,d}$ We may use the following 2-way branching strategy: Pick a variable x_i and a value a such that either setting $x_i = a$ or $x_i \neq a$ produces an instance which is found to be unsatisfiable by enforcing arc-consistency, and branch first to the "good" side. (We can replace arc consistency by other choices here, such as one-variable look-ahead plus forward checking.) If no such a combination exists, use any other scheme desired. We assume that singleton domains are always eliminated, including at startup. After startup every variable associated with a source node in G will have domain $\{2, \cdots, d\}$, and every variable x_t associated with a target node v_t would have been forced to take value 1, so the pebbling axioms for v_t with predecessors v_j and v_k in effect have become $\eta(x_j = a, x_k = b), a, b \in \{2, \cdots d\}$.

At any branching point, the chosen variable must be some x_i associated with a vertex v_i where both the variables x_j and x_k associated with predecessors of v_i have domain $\{2, \cdots, d\}$. Moreover, a_i will be 1. Setting $x_i = 1$ "falsifies" the literal $(x_1 = 1)$, so the pebbling axioms for v_i effectively become $\eta(x_j = a, x_k = b), a, b, \in \{2, \cdots, d\}$. These are inconsistent, which can easily be established with a search tree of size d^2. For the branch with $x_i \neq 1$, the pebbling axioms for v_i are satisfied and 1 is removed from the domain of x_i. Observe that there will always be a variable satisfying the criteria of our branching scheme, and the algorithm will effectively work its way from sources to a target, at each vertex efficiently removing 1 from the domain of a variable, and obtaining a trivial contradiction at the target. (The algorithm proceeds exactly as we would to pebble the graph). The total time required is certainly $O(d^2 n^2)$. The instances can be solved in about the same time by using repeated singleton arc consistency.

Separation of *tree-NG-RES* from *tree-C-RES* The implication graph contradictions $IMP_{G,S,T,d}$ have polynomial sized *tree-C-RES* refutations. Hence, they also separate *tree-NG-RES* from *tree-C-RES*.

Proposition 1. *For any circuit G with n vertices, tree-C-RES($IMP_{G,S,T,d}$) = $O(d^2 n)$.*

The refutations can be extracted from the algorithm above.

3.2 Separation of *tree-C-RES* from *C-RES*

Theorem 3. *There is an infinite family of instances $\{\mathcal{I}_n\}$ such that C-RES($\mathcal{I}_n$) = $O(n)$, and tree-C-RES($\mathcal{I}_n$) = $2^{\Omega(n/\log n)}$.*

Proof. A family of CNF formulas $\{\phi_n\}$ such that $|\phi_n| = O(n)$, $RES(\phi_n) = O(n)$ and $\textit{tree-RES}(\phi_n) = 2^{\Omega(n/\log n)}$ is given in [3]. Let $\mathcal{I}_n = \langle\{0,1\}, \Gamma_n\rangle$ be the transformation of ϕ_n to CSP, as follows. The variables of $\mathcal{I}\_n$ are just the variables in ϕ_n. For each clause C in ϕ_n, there is a nogood $\eta(\alpha)$ in Γ_n if and only if α is a minimal size truth assignment that makes C false. It is not hard to show that, if $\mathcal{I}$ is the transformation of ϕ as just described, then $RES(\phi) \leq \textit{C-RES}(\mathcal{I}) \leq 3 \cdot RES(\phi)$, and $\textit{tree-RES}(\phi) \leq \textit{tree-C-RES}(\mathcal{I}) \leq 3 \cdot \textit{tree-RES}(\phi)$ [10, 8]. The result follows.

3.3 Separation of *NG-RES* from *C-RES*

The family of CSP instances MGT'_n that separates *NG-RES* from *C-RES* is based on the unsatisfiable CNF formula GT_n introduced by [9]. For each $n \in \mathbb{N}$, GT_n encodes the negation of the fact that every loop-less transitive directed graph with n vertices and with no 2-cycles must have a source. The contradictory statement can be stated as a CNF formula containing the following clauses:

$$
\begin{array}{lll}
(1) & \overline{x_{j,j}} & j \in [n] \\
(2) & x_{i,j} \wedge x_{j,k} \rightarrow x_{i,k} & i,j,k \in [n],\ i \neq j \neq k \\
(3) & x_{i,j} \rightarrow \overline{x_{j,i}} & i,j \in [n],\ i \neq j \\
(4) & \bigvee_{i \in [n]} x_{i,j} & j \in [n]
\end{array}
$$

where $x_{i,j}$ takes value 1 if and only if there is an edge from i to j. The first three sets of clauses ensure that the graph is loop-less, transitive, and free of 2-cycles, respectively. The clauses in (4) assure that for each vertex j, there exists some vertex i such that there is an edge from i to j, i.e., there is no source. There are $O(n^3)$-size *RES* refutations of GT_n[14].

Bonet and Galesi [5] gave a modified version of GT_n, called MGT_n. For each $j \in [n]$, they introduce $n+1$ new variables $y_{0,j}, \ldots, y_{n,j}$ and replace the set of clauses (4) by:

$$
(4*) \quad \overline{y_{0,j}} \ \wedge \bigwedge_{i \in [n]} (y_{i-1,j} \vee x_{i,j} \vee \overline{y_{i,j}}) \ \wedge \ y_{n,j} \qquad j \in [n]
$$

The total number of variables is still $O(n^2)$ but MGT_n has constant width clauses. It is easy to see that we can derive the clauses in (4) from those in (4*) by resolving on the y variables and this takes $O(n^2)$ steps. Then, by applying

the $O(n^3)$-size refutation of GT_n, we obtain an $O(n^3)$-size *RES* refutation of MGT_n.

Our instances, MGT'_n, have the same set of variables as MGT_n but the domain for each variable is $D = \{1, 2, 3, 4\}$. If α is an assignment for MGT'_n, then

$$\alpha(x_{i,j}) = \begin{cases} 1 \text{ or } 2 & \text{means there exists an edge from } i \text{ to } j \\ 3 \text{ or } 4 & \text{means there is no edge from } i \text{ to } j. \end{cases}$$

So, every total assignment for the variables in MGT'_n corresponds to a directed graph with n vertices. To encode the contradictory statement, MGT'_n consists of the following nogoods:

$(1')$	$\eta(x_{j,j} = 1),\ \eta(x_{j,j} = 2)$	$j \in [n]$
$(2')$	$\eta(x_{i,j} = a,\ x_{j,k} = b,\ x_{i,k} = c)$	$i, j, k \in [n],\ i \neq j \neq k,$ $a, b \in \{1, 2\},\ c \in \{3, 4\}$
$(3')$	$\eta(x_{i,j} = a,\ x_{j,i} = b)$	$i, j \in [n],\ i \neq j,\ a, b \in \{1, 2\}$
$(4')$	for each $i \in [n]$, $\eta(y_{0,j} = 1),\ \eta(y_{0,j} = 2)$ $\eta(y_{i-1,j} = c,\ x_{i,j} = a,\ y_{i,j} = b)$ $\eta(y_{n,j} = 3),\ \eta(y_{n,j} = 4)$	$j \in [n],\ a, b \in \{1, 2\},\ c \in \{3, 4\}$

Theorem 4. *Any NG-RES refutation of MGT'_n must have size $2^{\Omega(n)}$.*

The proof of this is given in Section 5.

Theorem 5. *C-RES*$(MGT'_n) = O(n^3)$.

Proof. Derive the clauses $(\overline{x_{i,j} : a}\ \ \overline{x_{j,k} : b}\ \ x_{i,k} : 1\ \ x_{i,k} : 2)$, $i, j, k \in [n],\ i \neq j \neq k,\ a, b \in \{1, 2\}$, using the CNF encoding of (2') and the domain clauses of the x variables. Define

$$A(i, j, k) \stackrel{def}{=} \bigwedge_{a,b \in \{1,2\}} (\overline{x_{i,j} : a}\ \ \overline{x_{j,k} : b}\ \ x_{i,k} : 1\ \ x_{i,k} : 2).$$

Now, derive the clauses

$$P_m(j) \stackrel{def}{=} \bigvee_{\substack{i \in [m] \\ i \neq j}} X(i, j)$$

where

$$X(i, j) \stackrel{def}{=} (x_{i,j} : 1\ \ x_{i,j} : 2)$$

by resolving clauses in the CNF encoding of $(4')$ together with the domain clauses of the y and x variables, and unit clauses from (1'). Define $B(m, j)$ as

$$B(m, j) \stackrel{def}{=} \bigwedge_{a,b \in \{1,2\}} (\overline{x_{m,j} : a}\ \ \overline{x_{j,m} : b})$$

which is just the clauses in the CNF encoding of $(3')$. Now, for each $m < n$ and $j \leq m$, we can derive $P_m(j)$ from $P_{m+1}(j), A(i, m + 1, j)$, and $B(m + 1, j)$. Once we get $P_2(1)$ and $P_2(2)$, the empty clause can be derived in six steps. The *C-RES* derivation of $P_m(j)$ is of size $O(n)$. Therefore, we need $O(n^3)$ steps in total to derive the empty clause.

3.4 Separation Upper Bounds

Having provided some exponential separations between systems, it seems natural to ask how big the separations can be. For example, if we know that the smallest *NG-RES* refutation of a CSP instance $\mathcal{I}$ is of size S, then what is the upper limit for the size of the smallest *tree-NG-RES* refutation of $\mathcal{I}$ in terms of S? Here, for each of the separations we have provided, we give an upper bound on the best possible separation which might be obtained. These are only slightly larger than the lower bounds we give, so those results are nearly optimal.

Theorem 6. *For any n-variable CSP instance $\mathcal{I}$ with domain size $d \geq 2$,*

1. *tree-NG-RES($\mathcal{I}$) $= d^{O(\frac{d^2 S \log\log S}{\log S})}$, where $S =$ NG-RES($\mathcal{I}$);*
2. *tree-C-RES($\mathcal{I}$) $= 2^{O(S \log\log S/\log S)}$, where $S =$ C-RES($\mathcal{I}$),*
3. *tree-NG-RES($\mathcal{I}$) $= d^{O(nd^3 S \log\log S/\log S)}$, where $S =$ tree-C-RES($\mathcal{I}$).*
4. *NG-RES($\mathcal{I}$) $= 2^{O(S \log\log S/\log S)}$, where $S =$ C-RES($\mathcal{I}$),*

Proof. In [3] it is shown that for any unsatisfiable CNF formula ϕ, if $S = RES(\phi)$, then *tree-RES*$(\phi) = 2^{O(S \log\log S/\log S)}$, from which 2 follows easily. 1 can be proven by adapting the technique from [3] to *NG-RES*. 3 is obtained by using 1 and a direct simulation of *tree-C-RES* by *NG-RES*. 4 follows from 2 and the fact that *NG-RES* efficiently simulates *tree-C-RES*.

4 Lower Bounds for *tree-NG-RES*($IMP_{G,S,T,d}$)

This section comprises the proof of Theorem 1. The *tree-NG-RES* complexity of $IMP_{G,S,T,d}$ depends on the pebbling number of G. Roughly speaking, if G has large pebbling number, *tree-NG-RES* refutations of $IMP_{G,S,T,d}$ must be long.

Definition 2. *The* pebbling number *of T on a DAG $G = (V, E)$ from S, denoted $P_G(S, T)$, where $S, T \subseteq V$, is the minimal number of pebbles needed to pebble some vertex in T by following the rules below.*

1. *A pebble can be placed on a vertex in S.*
2. *A pebble can be removed from any vertex.*
3. *If a vertex is not in S, then it can only be pebbled if all its immediate predecessors have a pebble on them.*

Lemma 1 ([3]). *Let $G = (V, E)$ be a DAG. For any $v \in V$ and any sets $S, T \subseteq V$, $P_G(S, T) \leq \max\{P_G(S, T \cup \{v\}), P_G(S \cup \{v\}, T) + 1\}$.*

Proof. To pebble T from S, we can first pebble $T \cup \{v\}$ from S with $P_G(S, T \cup \{v\})$ pebbles. If some vertex in T is pebbled, then we are done. Otherwise, only v is pebbled. Leave the pebble on v and try to pebble T from $S \cup \{v\}$. This requires $P_G(S \cup \{v\}, T) + 1$ pebbles.

Note that for a DAG G with n vertices, $P_G(S,T) = O(n)$ since we can always use n pebbles and thus do not need to remove pebbles from vertices. What we are interested is a lower bound on the number of pebbles needed. A family of DAGs G_n with n vertices, each of in-degree 2 or 0, for which $P_{G_n}(S,T) = \Omega(n/\log n)$ where S and T are the sets of sources and targets in G_n, was given in [7].

The implication graph instance based on G_n is hard for *tree-NG-RES*. In particular, every *tree-NG-RES* refutation of $IMP_{G_n,S,T,d}$ must be of size $(d-1)^{\Omega(n/\log n)}$. We show this using a modified version of the game from [3], as follows. Let $\mathcal{I} = \langle [d], \Gamma \rangle$ be an unsatisfiable CSP instance. The game involves two players: Prover and Delayer. In each round, Prover picks a variable from $vars(\Gamma)$. Then, Delayer can choose 1 or *. If 1 is chosen, the variable is set to 1. Otherwise, Prover can pick a value from $\{2, \cdots, d\}$ and assign it to the variable. Delayer scores one point if he chooses *. The game ends when the current assignment falsifies at least one of the nogoods in Γ.

Here is a rough idea of the proof. We first show that any *tree-NG-RES* refutation of $IMP_{G,S,T,d}$ is of size at least exponential in the number of points Delayer can score. Then, we prove that there is a good strategy for Delayer to win at least $\Omega(P_G(S,T))$ points. So, every *tree-NG-RES* refutation of $IMP_{G,S,T,d}$ must be of size exponential to $\Omega(P_G(S,T))$. We call the above Delayer's strategy *superstrategy*.

Lemma 2. *For $\mathcal{I}$ an unsatisfiable CSP instance with domain size d, if $\mathcal{I}$ has a tree-NG-RES refutation of size S, then Prover has a strategy where Delayer can win at most $\lceil \log_{d-1} S \rceil$ points.*

Proof. Suppose $\mathcal{I}$ has a *tree-NG-RES* refutation π of size S. We will give a strategy which allows Prover to bound the number of points Delayer can win and show that as long as Prover follows the strategy, the following invariant will be maintained after each round: If p is the current points Delayer has scored, then there is a nogood N in π such that N is falsified by the current partial assignment and the sub-tree rooted at N in G_π is of size at most $S/(d-1)^p$.

At the beginning, Delayer has no points and the only nogood that is falsified is the empty nogood. So, the invariant holds. Consider the i-th round. Let p_{i-1} be the number of points Delayer has scored after the previous round and N_{i-1} be the nogood satisfying the invariant at the previous round. If N_{i-1} is a leaf in G_π, then N_{i-1} is a nogood in Γ that is falsified by the current partial assignment and hence the game ends. Otherwise, Prover picks the variable x which is resolved on to derive N_{i-1} from nogoods $N_1, N_2, \cdots, N_d$ in π. W.L.O.G., suppose $(x = 1) \in N_1, (x = 2) \in N_2$, and so on. If Delayer assigns 1 to x, then N_1 is falsified and it becomes the new nogood for the invariant. In this case, Delayer does not score any points and the sub-tree rooted at N_1 is obviously smaller than the one rooted at N_{i-1}. Thus, the invariant holds. If the Delayer chooses *, then Prover assigns x the value $j \in \{2, \cdots, d\}$ which will falsify the nogood N_j, among $N_2, \cdots, N_d$, with the smallest sub-tree. The sub-tree rooted at N_j is of size at most $1/(d-1)^{p_{i-1}+1}$, and the number of points Delayer has scored after this round is $p_{i-1} + 1$. Therefore, the invariant is maintained.

When the game halts, the size of the sub-tree is 1. If Delayer scores p points at the end of the game, then $1 \leq S/(d-1)^p$. This implies $p \leq \log_{d-1} S \leq \lceil \log_{d-1} S \rceil$. So, if Prover follows the above strategy, Delayer wins at most $\lceil \log_{d-1} S \rceil$ points.

Corollary 1. *For unsatisfiable CSP instance $\mathcal{I}$ with domain size d, if Delayer has a strategy which always scores r, then tree-NG-RES($\mathcal{I}$) $\geq (d-1)^{r-1}$.*

Proof. Suppose the Delayer has a strategy which always scores r points on $\mathcal{I}$. Toward a contradiction, suppose *tree-NG-RES*$(\mathcal{I}) < (d-1)^{r-1}$. Then, by Lemma 2, the Prover has a strategy where the Delayer can win at most $\lceil \log_{d-1}(d-1)^{r-1} \rceil = r-1 < r$ points. This contradicts that the Delayer can always scores r points.

The superstrategy for Delayer is simple. Before each game, Delayer sets $S' = S$ and $T' = T$. Then, in each round, if Prover asks about variable $x_i, i \in [n]$, Delayer responds as follows:

1. If $v_i \in T'$, assign 1 to the variable.
2. If $v_i \in S'$, respond *.
3. If $v_i \notin S' \cup T'$ and $P_G(S', T' \cup \{i\}) = P_G(S', T')$, assign the variable 1 and add v_i to T'.
4. If $v_i \notin S' \cup T'$ and $P_G(S', T' \cup \{i\}) < P_G(S', T')$, respond * and add v_i to S'.

We will prove that $P_G(S', T')$ can decrease by at most the number of points Delayer scores and it is at most 3 at the end of the game. This implies the superstrategy guarantees Delayer to earn at least $P_G(S, T) - 3$ points.

Lemma 3. *After each round, if Delayer has scored p points, then $P_G(S', T') \geq P_G(S, T) - p$.*

Proof. Let S'_i and T'_i be the sets S' and T' respectively in Delayers superstrategy after round i. Let p_i be the number of points Delayer has scored after round i. We show that the invariant $P_G(S'_i, T'_i) \geq P_G(S, T) - p_i$ will be maintained after each round. At the beginning, $p_0 = 0, S'_0 = S$ and $T'_0 = T$. So, $P_G(S'_0, T'_0) = P_G(S, T) - 0$ and the invariant holds. Now consider round i. For case 1, 2, and 3, $P_G(S'_{i-1}, T'_{i-1}) = P_G(S'_i, T'_i)$ and $p_i \geq p_{i-1}$. So, $P_G(S'_i, T'_i) = P_G(S'_{i-1}, T'_{i-1}) \geq P_G(S, T) - p_{i-1} \geq P_G(S, T) - p_i$. For case 4, $P_G(S'_{i-1}, T'_{i-1} \cup \{v\}) < P_G(S'_{i-1}, T'_{i-1})$, $p_i = p_{i-1}+1$, $S'_i = S'_{i-1} \cup \{v\}$, and $T'_i = T'_{i-1}$. By Lemma 1, we have $P_G(S'_{i-1} \cup \{v\}, T'_{i-1}) \geq P_G(S'_{i-1}, T'_{i-1}) - 1$. Hence, $P_G(S'_i, T'_i) = P_G(S'_{i-1} \cup \{v\}, T'_{i-1}) \geq P_G(S'_{i-1}, T'_{i-1}) - 1 \geq P_G(S, T) - p_{i-1} - 1 = P_G(S, T) - p_i$ Therefore, the invariant is maintained after each round.

Lemma 4. *At the end of the game, $P_G(S', T') \leq 3$.*

Proof. When the game ends, some nogood N must be falsified since $IMP_{G,S,T,d}$ is unsatisfiable. N cannot be a Source axiom for some source v_i because $v_i \in S \subseteq S'$ and thus it can only be assigned values from $\{2, \cdots, d\}$ through case 2. This

assignment does not violate the Source axiom. Similarly, N cannot be a Target axiom either. Hence, N must be a Pebbling axiom for some vertex v_k with predecessors v_i and v_j. To falsify N, x_k must be set to 1 and both x_i and x_j must be set to some values from $\{2, \cdots, d\}$. So, $v_k \in T'$ (via case 1 or case 3) and $v_i, v_j \in S'$ (via case 2 or case 4). Therefore, to pebble T' from S', we can first pebble v_i and v_j, then v_k. This only requires three pebbles.

Corollary 2. *Following the superstrategy described, Delayer can score at least $P_G(S,T) - 3$ points at the end of the game.*

Proof. This is an immediate consequence of Lemmas 3 and 4.

Lemma 5. *tree-NG-RES($IMP_{G,S,T,d}$) $= (d-1)^{\Omega(P_G(S,T))}$.*

Proof. Corollary 2 shows that Delayer has a superstrategy to score at least $P_G(S,T) - 3$ points on $IMP_{G,S,T,d}$. So, by Corollary 1, *tree-NG-RES*($IMP_{G,S,T,d}$) $\geq (d-1)^{P_G(S,T)-4}$. Hence, *tree-NG-RES*($IMP_{G,S,T,d}$) $= (d-1)^{\Omega(P_G(S,T))}$.

Proof. **(Theorem 1)** Let $d \geq 3$ be an integer. Let $\{G_n\}$ be an infinite family of circuits such that $|V(G_n)| = n$ and $P_{G_n}(S,T) = \Omega(n/\log n)$ where S and T are the sets of sources and targets in G_n [7]. By Lemma 5, we have *tree-NG-RES*($IMP_{G_n,S,T,d}$) $= (d-1)^{\Omega(P_G(S,T))} = (d-1)^{\Omega(n/\log n)}$.

5 Lower Bounds for *NG-RES*(MGT'_n)

This section comprises a proof of Theorem 4, which states that any *NG-RES* refutation of MGT'_n must have size $2^{\Omega(n)}$. The proof approach is inspired by [6]. We show that if there is a short *NG-RES* refutation of MGT'_n, then we can construct a narrow *RES* refutation of MGT_n, which contradicts the following property of MGT_n.

Theorem 7 ([5]). *Any RES refutation of MGT_n has width $\Omega(n)$.*

Definition 3. *A* restriction *for a CSP instance $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ forbids some variables to take some domain values. A restriction ρ is written as a set of variables with the forbidden values. For example, the restriction $\rho = \{x \neq 2, x \neq 3, y \neq 1\}$ disallows x to take 2 and 3, and y to take 1.*

Let $\rho = \{x_1 \neq a_1, x_2 \neq a_2, \cdots, x_k \neq a_k\}$ be a restriction. Define $N\lceil_\rho$ as the result of applying ρ to a nogood N where

$$N\lceil_\rho \stackrel{def}{=} (\cdots (N\lceil_{x_a \neq a_1})\lceil_{x_2 \neq a_2}) \cdots \lceil_{x_k \neq a_k}),$$

and for x a variable, $a \in \mathcal{D}(x)$,

$$N\lceil_{x \neq a} \stackrel{def}{=} \begin{cases} 1 & \text{if } (x = a) \in N \\ N & \text{otherwise.} \end{cases}$$

We define $\mathcal{I}\lceil_\rho \stackrel{def}{=} \langle \mathcal{D}\lceil_\rho, \Gamma\lceil_\rho \rangle$, where

$$\Gamma\lceil_\rho = \{N : N \in \Gamma \text{ and } N\lceil_\rho \neq 1\}$$
$$\text{vars}(\mathcal{I}\lceil_\rho) = \text{vars}(\mathcal{I})$$
$$\mathcal{D}\lceil_\rho(x) = \mathcal{D}(x) \setminus \{a : (x \neq a) \in \rho\} \qquad \text{for all } x \in \text{vars}(\mathcal{I}\lceil_\rho).$$

For $\pi = (N_1, \cdots, N_S)$ an NG-RES derivation, define $\pi\lceil_\rho$ to be $(N_1\lceil_\rho, \cdots, N_S\lceil_\rho)$, but with any $N_i\lceil_\rho$ that is identical to 1 removed. Note that $\pi\lceil_\rho$ is actually a subsequence of π.

Lemma 6. *If π is an NG-RES refutation of a CSP instance $\mathcal{I}$ and ρ is a restriction, then there is an NG-RES refutation of $\mathcal{I}\lceil_\rho$ of width at most $w(\pi\lceil_\rho)$.*

Proof. Let $\mathcal{I} = \langle \mathcal{D}, \Gamma \rangle$ be a CSP instance and $\rho = \{x \neq a\}$ be a unit restriction. Let π be an *NG-RES* refutation of $\mathcal{I}$. Transform $\pi\lceil_\rho$ inductively to an *NG-RES* refutation π' as follows. Consider a nogood N_i in $\pi\lceil_\rho$. $(x{=}a)$ must not appear in N_i since $N_i\lceil_{x\neq a} \neq 1$. If $N_i \in \Gamma$, then $N_i \in \Gamma\lceil_\rho$. (Note that $\pi\lceil_\rho$ is a subsequence of π.) Otherwise, N_i must be derived, in π, by resolving some previous nogoods $N_{i_1}, \cdots, N_{i_d}$ on some variable v. If $v \neq x$, then $(x = a)$ does not appear in any of $N_{i_1}, \cdots, N_{i_d}$ because $(x{=}a) \notin N_i$. So, $N_{i_1}, \cdots, N_{i_d}$ must be in $\pi\lceil_\rho$ and they can be resolved to derive N_i in $\pi\lceil_\rho$. If $v = x$, then there is a nogood $N_{i_a} \in \{N_{i_1}, \cdots, N_{i_d}\}$ such that $N_{i_a} = \eta(x = a, N_a)$ and thus $N_{i_a}\lceil_\rho$ is not in $\pi\lceil_\rho$ since $N_{i_a}\lceil_\rho = 1$. But, all the nogoods in $\{N_{i_1}, \cdots, N_{i_d}\} \setminus \{N_{i_a}\}$ are in $\pi\lceil_\rho$. So, we can resolve them on x, over the new domain of x, to get a sub-nogood of N_i, which is sufficient to produce the desired refutation. The general case for non-unit restriction follows easily.

Lemma 7. *If there is an NG-RES refutation of MGT'_n of size at most S, then there is a RES refutation of MGT_n of width at most w, for any $w > \log S$.*

Proof. Let π be an *NG-RES* refutation of MGT'_n of size at most S. Let $w > \log S$. Define that a nogood is wide if its width is greater than w. Define a random restriction ρ as follows. For each variable $v_{i,j}$, $v \in \{x, y\}$, ρ randomly picks a value a from $\{1, 2\}$ and a value c from $\{3, 4\}$, and restricts that $v_{i,j} \neq a$ and $v_{i,j} \neq c$. So, for every variable, a domain value is prohibited by ρ with probability $1/2$. We say that a restriction is bad if not all wide nogoods in π are set to 1 by ρ. A wide nogood would be set to 1 by ρ if there is some literal, $(x{=}a)$, in it such that $(x \neq a) \in \rho$. The probability that this is not the case is at most $1/2^w$. Since there is at most S nogoods in π, the probability that ρ is bad is at most $S/2^w$ which is less than 1 as we have $w > \log S$. Therefore, there must exist at least one good restriction which would set all wide nogoods in π to 1.

Apply a good restriction ρ to π. By Lemma 6, there is an *NG-RES* refutation π' of $MGT'_n\lceil_\rho$ of width at most w. After we apply ρ to MGT'_n, some initial nogoods disappear. For example, for each j, two of the nogoods in $(1')$ are set to 1 by ρ and thus not included in $MGT'_n\lceil_\rho$. Moreover, the domain size of each variable becomes 2.

Therefore, the CNF encoding of $MGT'_n\lceil_\rho$ consists of the following clauses:

$(1'')$ $(\overline{x_{j,j} : a_{j,j}})$ $\quad j \in [n]$

$(2'')$ $(\overline{x_{i,j} : a_{i,j}} \ \ \overline{x_{j,k} : a_{j,k}} \ \ \overline{x_{i,k} : c_{i,k}})$ $\quad i, j, k \in [n],\ i \neq j \neq k$

$(3'')$ $(\overline{x_{i,j} : a_{i,j}} \ \ \overline{x_{j,i} : a_{j,i}})$ $\quad i, j \in [n],\ i \neq j$

$(4'')$ $(\overline{y_{0,j} : b_{0,j}})$

$\bigwedge_{i \in [n]} \left(\overline{y_{i-1,j} : c_{i-1,j}} \ \ \overline{x_{i,j} : a_{i,j}} \ \ \overline{y_{i,j} : b_{i,j}} \right)$ $\quad j \in [n]$

$(\overline{y_{n,j} : d_{n,j}})$

Domain clauses: $(x_{i,j} : a_{i,j} \quad x_{i,j} : c_{i,j})$
$(y_{i,j} : b_{i,j} \quad y_{i,j} : d_{i,j})$

where each of $a_{i,j}$'s and $b_{i,j}$'s is equal to either 1 or 2 and each of $c_{i,j}$'s and $d_{i,j}$'s is equal to either 3 or 4.

Rename the variables $\overline{x_{i,j} : a_{i,j}}$, $\overline{x_{i,j} : c_{i,j}}$, $\overline{y_{i,j} : b_{i,j}}$, and $\overline{y_{i,j} : d_{i,j}}$ as $\overline{x_{i,j}}$, $x_{i,j}$, $\overline{y_{i,j}}$, and $y_{i,j}$, respectively. Now the constraint clauses of $\mathrm{CNF}(MGT'_n\lceil_\rho)$ are exactly the clauses in MGT_n and the *NG-RES* derivation steps

$$\frac{\eta(x_{i,j} = a_{i,j},\ N_1) \quad \eta(x_{i,j} = c_{i,j},\ N_2)}{\eta(N_1, N_2)}\ x_{i,j} \in \{a_{i,j}, c_{i,j}\} \quad \text{and} \quad \frac{\eta(y_{i,j} = b_{i,j},\ N_1) \quad \eta(y_{i,j} = d_{i,j},\ N_2)}{\eta(N_1, N_2)}\ y_{i,j} \in \{b_{i,j}, d_{i,j}\}$$

in π' can be transformed into the following *RES* derivation steps

$$\frac{(\overline{x_{i,j}} \quad X_1) \quad (x_{i,j} \quad X_2)}{(X_1 \ X_2)} \quad \text{and} \quad \frac{(\overline{y_{i,j}} \quad X_1) \quad (y_{i,j} \quad X_2)}{(X_1 \ X_2)}\ .$$

The resulting *RES* refutation has the same width as π'. Hence, there is a *RES* refutation of MGT_n of width at most w.

Proof. **(of Theorem 4)** Let π be an *NG-RES* refutation of MGT'_n. Let S be the size of π. Pick $w = \log S + \epsilon,\ \epsilon > 0$. It follows from Lemma 7 that MGT_n has a *RES* refutation π' of width at most $\log S + \epsilon$. We know that any *RES* refutation of MGT_n must have width $\Omega(n)$ (Theorem 7). Therefore, $\log S + \epsilon \geq \Omega(n)$, and thus $S \geq 2^{\Omega(n)}$. Hence, any *NG-RES* refutation of MGT'_n must be of size $2^{\Omega(n)}$.

6 Conclusion and Future Work

We have shown that 2-way branching is much more powerful than d-way branching, for backtracking. It remains to establish an efficient strategy for 2-way branching with learning for the instances separating *NG-RES* from *C-RES*, to establish the analogous fact for the case with learning. The question of whether nogood learning algorithms are as powerful as these proof systems is an important open problem.

The algorithm suggested for efficiently solving the instances which separate *tree-NG-RES* and *tree-C-RES* are simple enough that they certainly will be faster, at least for large enough instances, than any d-way branching algorithm. However, developing good heuristics which take advantage of the extra power of 2-way branching in practical algorithms is an important remaining task.

References

[1] Andrew B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results.* PhD thesis, University of Oregon, 1995.

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

[3] Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near-optimal separation of treelike and general resolution. Technical Report TR01-005, Electronic Colloquium on Computational Complexity (ECCC), 2000.

[4] M. L. Bonet, J. L. Esteban, N. Galesi, and J. Johansen. Exponential separations between restricted resolution and cutting planes proof systems. In *Proc. of the 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS'98)*, pages 638–647. IEEE Press, 1998.

[5] M. L. Bonet and N. Galesi. A study of proof search algorithms for resolution and polynomial calculus. In *Proc. 40th Symposium on Foundations of Computer Science*, pages 422–432, 1999.

[6] J. Buresh-Oppenheim, D. Mitchell, and T. Pitassi. Linear and negative resolution are weaker than resolution. Technical Report TR01-074, Electronic Colloquium on Computational Complexity (ECCC), 2001.

[7] J.R. Celoni, W.J. Paul, and R.E. Tarjan. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977.

[8] Cho Yee Joey Hwang. A theoretical comparison of resolution proof systems for csp algorithms. Master's thesis, Simon Fraser University, 2004.

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

[10] David G. Mitchell. *The Resolution Complexity of Constraint Satisfaction*. PhD thesis, University of Toronto, 2002.

[11] David G. Mitchell. Resolution and constraint satisfaction. In *Lecture Notes in Computer Science, LNCS 2833*, pages 555–569. Springer, 2003.

[12] V. Park. An empirical study of different branching strategies for constraint satisfaction problems. Master's thesis, University of Waterloo, 2004.

[13] B. M. Smith and P. Sturdy. An empirical investigation of value ordering for finding all solutions. Presented at the ECAI 2004 workshop on Modelling and Solving Problems with Constraints.

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