Crossword Puzzles: A Case Study in Compute-Intensive Meta-Reasoning*

James J. Lu[1], Jeffrey S. Rosenthal[2], and Andrew E. Shaffer[3]

[1] Bucknell University, Lewisburg, PA 17838, U.S.A. jameslu@bucknell.edu
[2] University of Toronto, Toronto, Canada M5S 3G3 jeff@math.toronto.edu
[3] Bucknell University, Lewisburg, PA 17838, U.S.A. ashaffer@bucknell.edu

Abstract. A new technique for improving the efficiency of propositional reasoning procedures is presented. The meta-search procedure, ND, is parameterized by a search procedure P and a real number for controlling the way in which P is applied to the given problem. Experiments using SATO [Zha97] on the domain of Crossword Puzzle Construction (CPC) illustrate the potential for ND.

KEYWORDS: automated reasoning, search, threshold phenomena, control

1 INTRODUCTION

Finding effective strategies for controlling propositional inference systems continues to be a challenging research issue. Control strategies such as set-of-support ([WRC65]) and linear ([Lov70]) restrictions represent a limited form of *informed search* [RN95], whereby restriction and guidance of search is based on knowledge of the underlying problem domain. For the most part, current research directions in the field have continued to rely on knowledge-intensive techniques for restricting as well as for directing reasoning programs [MW92,Bun93,WP99].

On the other hand, promising compute-intensive search techniques have recently been applied to successfully solve many problems that are beyond the current capabilities of knowledge-intensive reasoning procedures [LMS92b,GSK98]. These compute-intensive techniques, as described by Selman, attack the inherent combinatorics of problems from first principles, with little or no domain-specific knowledge.

In this paper, we describe a compute-intensive meta-reasoning technique for improving the efficiency of propositional reasoning procedures. Informally, the new technique, ND, assumes the existence of a base procedure P and a method D for reducing a problem instance Ω to a related subproblem. ND first finds the solutions of the subproblem $D(\Omega)$ using P. It then directs the search for a solution of Ω by, using P repeatedly, attempting to extend the solutions of $D(\Omega)$. The interesting idea here is, ND does not contain any knowledge of the problem domain. Its only ability to affect the performance of P is based on a real number, which indicates the percentage of solutions of $D(\Omega)$ to consider in each repetition.

As preliminary experiments, we explore the effectiveness of ND on randomly generated problem instances of Crossword Puzzle Construction (CPC). In spite of its long

* **Acknowledgments.** James Lu and Andrew Shaffer are supported in part by the National Science Foundation under Grant No. CCR9731893. Jeffrey Rosenthal is supported in part by a research grant from the Natural Science and Engineering Research Council of Canada.

history (dating back to before 1976 [Maz76]) as a test bed for automated reasoning techniques, CPC still presents considerable challenges to both complete and stochastic search procedures [Kon94]. We demonstrate, for two complete search strategies implemented in SATO [Zha97], substantial performance improvement for the computationally most difficult problem instances of CPC. We provide mathematical analysis which shows that the gain in efficiency occurs with minimal loss in completeness.

In Section 2, the meta-search procedure ND is formalized. We then discuss the preliminary result of applying ND to the problem of Crossword Puzzle Construction in Section 3.

2 THE META-PROCEDURE ND

Given a problem instance Ω (represented as a set of propositional clauses) and a propositional reasoning procedure P for determining the models of Ω, consider a decomposition of Ω into a smaller but related subproblem $\mathcal{D}(\Omega)$.[1] Suppose $ss(\mathcal{D}(\Omega))$ is the set of all models of $\mathcal{D}(\Omega)$. Then, for each $s_0 \in ss(\mathcal{D}(\Omega))$, s_0 can be used as additional constraints to prune the search for a solution to Ω by the procedure P. Now in general, the time it takes to discover the models of $\mathcal{D}(\Omega)$ using P is considerably less than the time it takes to discover a model of Ω. Thus, the overall time for solving Ω may potentially be improved. The idea is described more formally below in the procedure SD.

Input: a problem instance Ω, a propositional reasoning procedure P, and a decomposition method $\mathcal{D}$
Output: a model of Ω or the answer 'no'

1. Compute $\mathcal{D}(\Omega)$.
2. Generate $ss(\mathcal{D}(\Omega))$ using P.
3. **while** $ss(\mathcal{D}(\Omega))$ is non-empty
   (a) Remove a model s_0 from $ss(\mathcal{D}(\Omega))$.
   (b) Let $\Omega^+ = \Omega \cup s_0$.
   (c) If $P(\Omega^+)$ is solvable, stop and return the solution.
4. Stop and return 'no'.

Procedure SD(Ω,P,$\mathcal{D}$)

If the time it takes to solve a problem instance Ω using a procedure Γ is represented by the function $t_{\Gamma(\Omega)}$, then our hope from SD is that, for a sufficiently large number of instances Ω over the problem domain, $t_{SD(\Omega,P,\mathcal{D})}$ will be smaller than $t_{P(\Omega)}$.

In the worst-case, the running time for SD is

$$d + tss + \Sigma_{s_0 \in ss(\mathcal{D}(\Omega))}(t_{P(\Omega \cup s_0)}),$$

where d and tss are the times required to decompose Ω and generate $ss(\mathcal{D}(\Omega))$, respectively.

[1] The exact nature of the decomposition is problem dependent.

Unfortunately, experiments suggest that, on average, the time required by SD is in fact larger than the time needed to simply run P on the original problem instances. The qualitative explanation is that the number of times Step 3 is repeated is in general large (due to the size of $ss(D(\Omega))$).

On the other hand, we will show that if the solution space of $\mathcal{D}(\Omega)$ is explored not one element at a time, but a group at a time, then the average time required for finding a solution to Ω can be improved dramatically, in some cases outperforming P by a large margin. The idea is described more precisely in the procedure ND below. It is adapted from SD to allow for groups of models in $ss(\mathcal{D}(\Omega))$ to be added as constraints to Ω. For correctness, additional constraints are necessary since models of $ss(\mathcal{D}(\Omega))$ are not pairwise consistent. The constraints ensure that given a set of models $s_0, ..., s_k$, one and only one of the s_i's can be true, $0 \leq i \leq k$. We denote such a set of constraints mutex$(s_0, ..., s_k)$.

Input: a problem instance Ω, a propositional reasoning procedure P, a decomposition method $\mathcal{D}$, and a number $n \in (0, 100]$
Output: a model of Ω or 'no'

1. Compute $\mathcal{D}(\Omega)$.
2. Generate $ss(\mathcal{D}(\Omega))$ using P. Let m denote the cardinality of $ss(\mathcal{D}(\Omega))$.
3. **while** $ss(\mathcal{D}(\Omega))$ is non-empty
   (a) Remove models $s_0, ..., s_k$ from $ss(\mathcal{D}(\Omega))$, where $k = min(m', \frac{mn}{100})$ and m' is the current size of $ss(\mathcal{D}(\Omega))$.
   (b) Let $\Omega^+ = \Omega \cup$ mutex$(s_0, ..., s_k)$.
   (c) If Ω^+ is solvable using P, then stop and return the solution.
4. Stop and return 'no'.

Procedure ND(Ω,P,$\mathcal{D}$,n)

The parameter n controls the number of solutions of $\mathcal{D}(\Omega)$ to be added to Ω at Step 2(b). It is specified as a percentage. We call n the *control variable*.

ND is a meta-reasoning procedure because it does not perform any inference itself, and its effectiveness on a particular problem is intimately tied to the effectiveness of the given procedure P. Experiments suggest, however, that regardless of the procedure provided, there is a range of values for n for which the average time it takes for ND(Ω,P,$\mathcal{D}$, n) to complete is faster than $P(\Omega)$.

3 CROSSWORD PUZZLE CONSTRUCTION

The problem of CPC can be stated as follows: given a finite set of words W and an $N \times N$ grid for which some squares in the grid are shaded in while others are open for letters, can all the open squares be filled with letters such that every horizontal and vertical maximal group of adjacent letters forms a word from W? We further require that no word from W can appear in the grid more than once.[2]

The general problem of CPC is complex and a complete understanding of the nature of the problem is beyond the scope here. To make the problem instances more amenable to analysis, we considered the modified problem of Open CPC (OCPC) — CPCs with the assumption that no square is shaded.

For the experiments, we fix an $N \times N$ open grid and an alphabet of size A. Strings of length N are randomly generated from elements of the alphabet to form the set W. For a particular W, the puzzle is encoded into a set of propositional clauses as follows.[3] Each propositional variable in the clauses represents a triple `(r,c,let)`, which indicates that the letter `let` is placed in the grid position `(r,c)`. Hence, the total number of variables equals N^2A.

There are clauses to constrain the placements of the letters on the squares. We do not elaborate on their details.

3.1 APPLYING SATO TO OPEN CPC

Experiments with randomly generated OCPC instances exhibit the phase transition phenomenon found in other combinatorial search problems [CKT91,LMS92a]. For instance, phase transitions for randomly generated open 3×3 puzzles with alphabet sizes ranging from 2 to 12 are shown in Figure 1. Various properties of the open 3×3

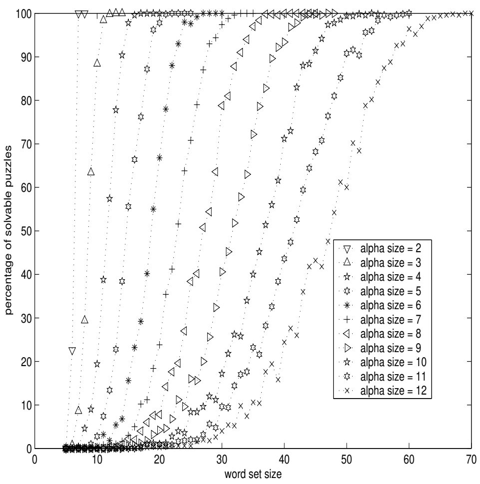


Fig. 1. Open 3×3 CPC

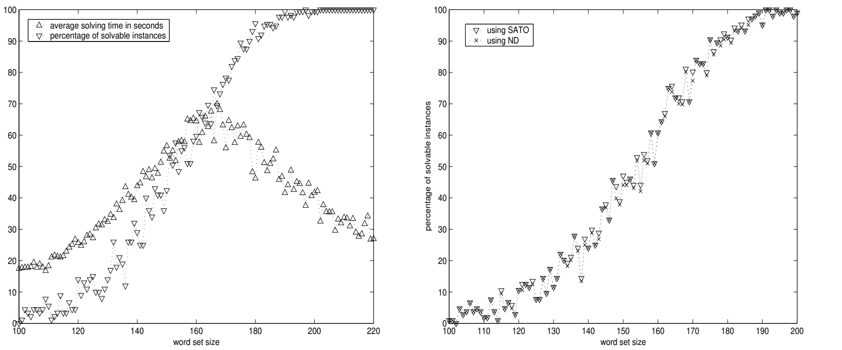
puzzles can be investigated based on the data. As an example, for a given alphabet, the word set size needed to achieve 50% solvable instances can be modeled by the

[2] The no duplicate word constraint is not in the formulation of the CPC problem in [GJ79]. We add this requirement since we are also interested in understanding, as much as possible, the nature of real-world crossword puzzle.

[3] Instances of CPC can be encoded as propositional clauses in different ways [Kon94].

equation $|A|^{1.516} + 3.8$. Projected to a 26 letter alphabet — the size of the English Alphabet — yields an expected word set size of 143.67.[4]

Consistent with other explorations into combinatorial search problems, the computationally most difficult problem instances of CPC occur during the phase transition. These problem instances are the *critically constrained* ones. That is, they occur at a critical ratio of variables to clauses. Figure 2.A plots the phase transition for open 4×4, 12 letter alphabet puzzles along with the average solving time (in seconds) using SATO. The data are based on over 100 trials for each word set size.[5]



A. Phase Transition B. Invariability in satisfiability using Chp

Fig. 2. Open 4×4 puzzles over $A = 12$.

3.2 DECOMPOSING OPEN CPC

A number of possibilities exist for decomposing an OCPC into related subproblems. The one we consider can be described as follows. As before, given an Open $N \times N$ CPC instance Ω, we denote W the set of words. Then, the decomposition is the puzzle whose grid is the upper left-hand $M \times M$ grid of Ω, $M < N$, and whose word set is obtained from W by removing the last $N - M$ letters from each word in W. Since we consider only solutions that do not contain duplicate words, if two or more words in W begin with the same M letters, only one copy of the reduced word is kept. We will denote this method of decomposition for Open CPC Chp_M.

There are two issues to consider. The first is to demonstrate a speed up in average solving time for $\mathrm{ND}(\Omega, \mathrm{SATO}, Chp_M, n)$ over $\mathrm{SATO}(\Omega)$ for some $n \in (0, 100]$. The second is to analyze how the use of Chp_M as a method of decomposing problem instances affects completeness. We explore the second issue in the next section. We show that although the use of Chp_M does not guarantee completeness, the probability of finding a solution (when one exists) is quite high for even relatively simple puzzles.

[4] This says, given an open 3×3 grid and 3-letter words formed over a 26 letter alphabet, we need roughly 144 words to have a 50% chance of constructing a puzzle.

[5] **Remark.** In spite of their relative small sizes, encodings for 4×4, 12 letter alphabet puzzles produce propositional representations of non-trivial sizes ranging from 8000 to 10000 clauses over 192 variables, depending on the word set size. As in [Kon94], most of these instances are quite difficult for GSAT [LMS92b].

In the section on Solving Time Improvement, we present and discuss the speed ups for a few Open CPC by applying ND. Together, the two results demonstrate the utility of ND over CPC and argue for ND's potential for other search problems.

3.3 CORRECTNESS ISSUES

As mentioned, two or more words of W may map onto the same word in the decomposition through Chp_M. There are two consequences of this effect. First, $Chp_M(\Omega)$ may have no solution even though solutions exist for Ω. Second, the models of $Chp_M(\Omega)$ do not necessarily extend to any model of Ω. In spite of these possible inconsistencies between the solution spaces of a puzzle and its decomposition, such inconsistencies appear to occur infrequently over the space of randomly generated puzzle instances. Below we provide some analysis.

Mathematical bounds for Open CPC Fix an $N \times N$ open square grid, an alphabet size $A \geq 2$, and a number W of distinct words (to be chosen uniformly at random from the A^N possible words). The expected (i.e., average) number of solutions, E, satisfies the following.

Theorem 1. The quantity E satisfies

$$E = FT/P$$

where F satisfies $A^{N^2}\left[1 - \binom{2N}{2} A^{-N}\right] \leq F \leq A^{N^2}$, where $T = \binom{A^N - 2N}{W - 2N}$, and where $P = \binom{A^N}{W}$. Here $\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$ is the usual binomial coefficient.

Remark. This theorem provides bounds on the "slugging percentage" or "expected value", E, as a function of N, A, and W. The bounds are not 100% sharp because of the uncertainty in the range of F. However they get more and more sharp, on a relative scale, as either the grid size N or the alphabet size A increases. Even for moderate values of N and A they provide fairly useful bounds.

Consider a numerical example, if the grid size is $N = 3$, and the alphabet size is $A = 26$, and if the number of words is $W = 144$ (to give, as shown previously, approximately a probability of 0.5 of having at least one solution), then this Theorem says that the expected number of solutions, E, satisfies $1.47771 \leq E \leq 1.47897$.

We thus see that the theorem gives very tight bounds on E. Furthermore the resulting values are reasonable; they indicate when there is a probability of about 0.5 of having at least one solution, the average number of multiple solutions is about 3.

Extensions of Sub-solution The primary question we are interested in answering is, suppose we fix A and W, and let $M < N$ be two positive integers. Then, of all possible solutions of Open CPC on the $N \times N$ grid, what expected fraction of them are extensions of solutions on the $M \times M$ grid formed by considering just the upper-left-hand corner of the $N \times N$ grid? (The point is that, to be extensions of $M \times M$ solutions, they must have their upper-left $2M$ words be distinct somewhere in the first M letters.)

We have the following result.

Theorem 2. Of all solutions of the $N \times N$ Open CPC problem, the expected fraction which are extensions of solutions of the $M \times M$ upper-left-hand-corner problem is at least

$$1 - \binom{2M}{2} A^{-M} - \left[\binom{2N}{2} - \binom{2M}{2}\right] A^{-N}.$$

(This bound does not depend on the value of W, though of course we need $W \geq 2N$ for the statement to make sense).

For example, if there are $A = 18$ letters in the alphabet, and $N = 4$ is the size of the grid, and $M = 3$ is the size of the sub-grid, then the fraction described in Theorem 2 is at least 0.997304. Even if $A = 8$ (keeping $N = 4$ and $M = 3$), the fraction is at least 0.967529. On the other hand, if $A = 4$ then the bound of Theorem 2 is only 0.714844, and for $A = 3$ is it just 0.283951. (By comparison, if $N = 8$ and $M = 7$ with $A = 3$, then the fraction is a healthy 0.95397.)

Clearly, as either $A \to \infty$ with N, M fixed, or as $N, M \to \infty$ with A fixed, the fraction of Theorem 2 converges to 1. Figure 2.B shows experimental results confirming the relative small loss in correctness when using *Chp*. The data are based on an open 4×4 grid, 12 letter alphabet puzzles when decomposition is to its upper left-hand 2×2 corner. In the figure, the $\triangledown$ symbols denote the percentage of solvable instances, computed by SATO, for each word set size between 100 and 220 (as shown previously in Figure 2.A), while the `x` symbols represent the percentage of solvable instances over the same puzzles, computed by ND, by first solving the upper left-hand 2×2 subproblems. As can be seen from the figure, for each word set size, the ratio of the `x` value to the $\triangledown$ value adheres closely to the value that is bounded by Theorem 2 which, for the given parameters, is at least 0.9573.

3.4 SOLVING TIME IMPROVEMENT

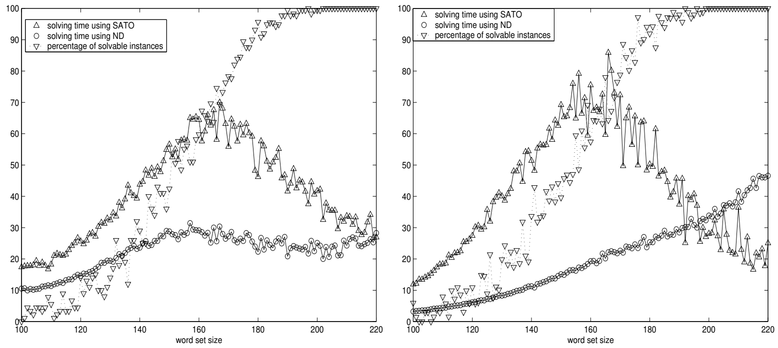
Experiments were conducted to compare the solving times on a variety of puzzle sizes using SATO and ND. Recall first that when ND is applied, a value for the control variable n must be specified. This value indicates the number of subsolutions of the decomposed problem to be inserted simultaneously into the overall problem. Different choices in the value of the control variable result in different solving time. We discuss experiments with the open 4×4 grid, 12 letter alphabet puzzle, shown previously in Figure 2.A. Results of other experiments will be summarized.

For the 4×4 grid, 12 letter alphabet puzzle, we chose to decompose each problem instance to the upper left-hand 2×2 grid as well as to the upper left-hand 3×3 grid. From Figure 2.B, we know that satisfiability is preserved to a large extent even in the case of the 2×2 decomposition. To determine the value for the control variable of ND, a small sample of values was attempted for a number of selected word set sizes using the 3×3 decomposition. A plot of these values and some curve fitting yielded the following equation which was used for computing the value of n.[6]

$$100(\frac{106.93}{W} + 0.0031724 \cdot W - 1.14163) \tag{1}$$

[6] Curve fitting as a way to determine the optimal value for the control variable was done only on 4×4 puzzles. For other puzzles, we fixed the value of the control variable to a constant and still obtained good speed ups, as will be discussed later.

Thus, for instance, given 100 words in W, the value for n is set to $100(\frac{106.93}{100} + 0.0031724 \cdot 100 - 1.14163)$, or approximately 24. This indicates that in each iteration of Step 3 of ND, 24 percent of the solutions for the decomposition will be inserted simultaneously, along with the constraint that they are mutually exclusive, into the original problem for pruning the search of SATO. Hence, the loop of ND will be repeated at most 5 times. In contrast, if W contains 160 words, $n = 3.84$ and the loop will be repeated at most 26 times.



A. 2×2 decomposition　　B. 3×3 decomposition

Fig. 3. ND vs. SATO

Based on Equation 1 for the control variable, ND was applied to the open 4×4 grid, 12 letter alphabet puzzles, for word sizes ranging from 100 to 220. Figure 3.A compares the solving time required by ND to the solving time used by SATO (which is the same as what's shown in Figure 2.A) when decomposition is to the upper left-hand 2×2 grid.[7] Figure 3.B on the other hand, compares the solving time for the two search techniques when decomposition is to the upper left-hand 3×3 grid.[8]

Some observations about Figure 3.A. When problem instances are easily determined to be unsatisfiable, ND achieves a speed up of 40 to 50 percent of the time required by SATO. Similar improvement is obtained when problem instances approach 100 percent satisfiable.

For the computationally most difficult instances (word set sizes between 150 and 180), the speed up increases to about 60 percent of the time required by SATO. Noteworthy also is that the increase in solving time by ND is relatively stable in the word set size of the problem. While a direct application of SATO shows a dramatic increase and variability in time requirements during the phase transition, the increase rate in ND almost appears to be unaffected by the same threshold phenomenon that affects SATO. The unpredicatability of complete search procedures has been recognized as a consequence of the "heavy-tailed cost distributions" [GSCK99]. In this respect, the behavior of ND resembles the behaviors of randomized complete algorithms [GSK98].

[7] Notwithstanding the good result displayed in Figure 3.A, the time improvement achieved by ND for the 2×2 decomposition is not necessarily optimal since Equation 1 was developed based on data gathered for the 3×3 decomposition. We will revisit this point later.

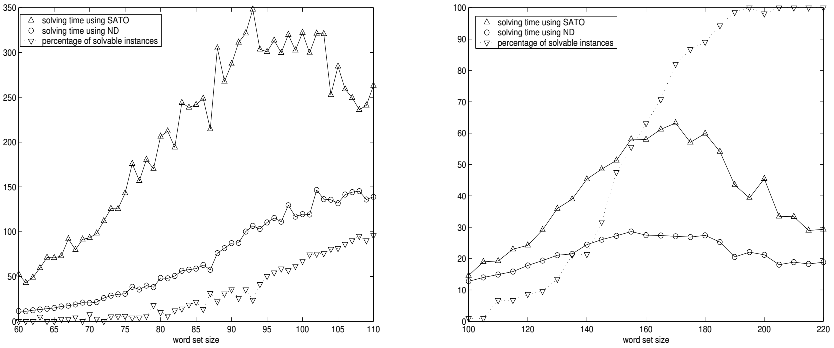
[8] Note the total solving time using SATO in Figure 3.B is slightly different from those shown in Figures 2.A and 3.A because the experiments were conducted on a different machine.

The data for the 3×3 decomposition shown in Figure 3.B provides an even more dramatic demonstration of the apparent independence of ND from the threshold phenomenon.

3.5 OTHER EXPERIMENTS AND CURRENT RESEARCH

Highlights of other experiments are summarized here.

Puzzle Independence. Experiments were conducted for OCPCs of varying sizes and speed ups similar to the ones for open 4×4, 12 letter alphabet were obtained. Figure 4.A illustrates speed ups between 40 to 80 percent when ND is applied to open 5×5, 6 letter alphabet puzzles by decomposing to the 3×3 upper left-hand corners and using a value of 1.5 for the control variable. For these parameters, Theorem 2 tells us that at least 92.67 percent of all solutions to the 5×5 grid are extensions of solutions of the 3×3 upper-left-hand grid.



A. Open 5×5, 6 letter alphabet puzzles B. ND vs. SATO (using control value 1.5)

Fig. 4. Other Experiments

Control Variable Variance. As mentioned, varying the value of the control variable can affect the amount of speed up. In fact, Equation 1 is tailored specifically to the 3×3 decomposition for the open 4×4 grid. Therefore, the speed up obtained via ND on the 2×2 decomposition is not necessarily optimal. Figure 4.B demonstrates that fixing the control value to a constant 1.5 for the 2×2 decomposition gives slightly different performance for ND. Clearly, the determination of an optimal control value is an important research topic for gaining further insights into the behavior of ND.

Search Strategy Independence. SATO has several different search strategies built in, including a form of intelligent backjumping (IB). A number of experiments were conducted using both the default search strategy of SATO as well as IB, and in each case, solving time was improved. This demonstrates that the effectiveness of ND is, apparently, independent of the type of search strategy employed in the base procedure.

Other Problem Domains. Although the data for CPC provides evidence for ND as a promising technique for controlling propositional reasoning systems, experimentation with other problem domains will certainly be necessary for validation. We are currently exploring the application of ND to graph colorability as well as other problems, and we will report these results in the full version of the paper.

References

[Bun93] A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and A. Smaill, Rippling: A Heuristic for Guiding Inductive Proofs, *Artificial Intelligence*, 62, 1993. 185–253.

[CKT91] P. Cheeseman, B. Kanefsky, and W. Taylor, Where the Really Hard Problems Are. *Proceedings of IJCAI-91*, San Mateo, CA, 1991, 331–337.

[GJ79] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Francisco, 1979.

[GSCK99] C. Gomes, B. Selman, Nuno Crato, and H. Kautz, Heavy-Tailed Phenomena in Satisfiability and Constraint Satisfaction Problems, to appear in *Journal of Automated Reasoning*.

[GSK98] C. Gomes, B. Selman, and H. Kautz, Boosting Combinatorial Search Through Randomization, *Proceedings of AAAI* (AAAI-98), Madison, WI, July 1998.

[Kon94] K. Konolige, Easy to be Hard: Difficult Problems for Greedy Algorithms. *Proceedings of KR-94*, Bonn, Germany, 1994, 374–378.

[LMS92a] H. Levesque, D. Mitchell, and B. Selman, Hard and Easy Distributions of SAT Problems. *Proceedings of AAAI* (AAAI-92), San Jose, CA, July 1992, 459–465.

[LMS92b] H. Levesque, D. Mitchell, and B. Selman, A New Method for Solving Hard Satisfiability Problems. *Proceedings of AAAI* (AAAI-92), San Jose, CA, July 1992, 440–446.

[Lov70] D. Loveland, A Linear Format for Resolution, *Proceedings of the IRIA Symposium on Automatic Demonstration*, Springer, 1970, 147–162.

[Maz76] L.J. Mazlack, Computer Construction of Crossword Puzzles using Precedence Relationships. *Artificial Intelligence*, 7, 1976, 1–19.

[MW92] W. McCune and L. Wos. Experiments in Automated Deduction with Condensed Detachment. *Proceedings of CADE*, Springer, 1992.

[RN95] S. Russell, and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Englewood Cliffs, 1995.

[WP99] L. Wos and G. Pieper, The Hot List Strategy, *J. of Automated Reasoning*, 22(1), 1999, 1–44.

[WRC65] L. Wos, G.A. Robinson, and D.F. Carson, Efficiency and Completeness of the Set of Support Strategy in Theorem Proving, *J. ACM*, 12(4), 1965, 536–541.

[Zha97] H. Zhang, SATO: An Efficient Propositional Prover. *Proceedings of CADE*, Spring, 1997.