Solving the Really Hard Problems with Cooperative Search

Tad Hogg and Colin P. Williams

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304, U.S.A.
Hogg@parc.xerox.com, CWilliams@parc.xerox.com

Abstract

We present and experimentally evaluate the hypothesis that *cooperative* parallel search is well suited for hard graph coloring problems near a previously identified transition between under- and overconstrained instances. We find that simple cooperative methods can often solve such problems faster than the same number of independent agents.

Introduction

Many A.I. programs involve search to solve NP hard problems. While intractable in the worst case, of more relevance to many applications is their behavior in typical situations. In fact, for many classes of such problems, most instances can be solved much more readily than might be expected from a worst case analysis. This has led to recent studies to identify characteristics of the relatively hard instances. In particular, observations [Cheeseman et al., 1991, Mitchell et al., 1992] and theory [Williams and Hogg, 1992b, Williams and Hogg, 1992a] indicate that constraint-satisfaction search problems with highest cost (fastest growing exponential scaling) occur near the transition from under- to overconstrained problems. These transitions, becoming increasingly sharp as problems are scaled up, are determined by values of easily measured "order" parameters of the problem, and are analogous to physical phase transitions as in percolation. The transition regions are also characterized by high variance in the solution cost for different problem instances, and for a single instance with respect to different search methods or a single nondeterministic method with e.g., different initial conditions or different tie-breaking choices made when the search heuristic ranks some choices equally. Structurally, these hard problems are characterized by many large partial solutions, which prevent early pruning by many types of heuristics. This phenomenon is also conjectured to appear in other types of search problems.

Can these observations be exploited in practical search algorithms? One immediate application is to use the order parameters to estimate the difficulty of alternate problem formulations as an aid in deciding which approach to take. Another use is based on the observation of high variance in solution cost for problems near the transition region. Specifically, there have been many studies of the benefit of running several methods independently in parallel and stopping when any method first finds a solution [Fishburn, 1984, Helmbold and McDowell, 1989, Pramanick and Kuhl, 1991, Kornfeld, 1981, Imai et al., 1979, Rao and Kumer, 1992, Mehrotra and Gehringer, 1985]. Since the benefit of this approach relies on variation in the individual methods employed, the high variance seen in the transition region suggests it should be particularly applicable for hard problems [Cheeseman et al., 1991, Rao and Kumer, 1992].

Another possibility is to allow such programs to exchange and reuse information found during the search, rather than executing independently. If the search methods are sufficiently diverse but nevertheless occasionally able to utilize information found in other parts of the search space, greater performance improvements are possible. Such "cooperative" methods have been studied in the context of simple constraint satisfaction searches [Clearwater et al., 1991, Clearwater et al., 1992]. In these cases, cooperative methods were observed to give the most benefit precisely for those problems with many large partial solutions that could not be pruned. This was the case even though the information exchanged was often misleading in the sense of not being part of any solution. While this work used fairly simple search methods, it suggests that cooperative search may be useful for much harder problems employing sophisticated search heuristics.

These observations lead us to conjecture that a mixture of diverse search methods that share information will be particularly effective for problems in the transition region. In this paper we test this conjecture experimentally for graph coloring, a particular class of NP-complete problems for which an appropriate order parameter, average connectivity, and the location of the transition region have been empirically determined [Cheeseman et al., 1991]. We also address some practical issues of sharing information, or exchanging "hints", among sophisticated heuristic search methods, which should allow these results to be extended readily to other constraint satisfaction problems.

Hard Graph Coloring Problems

The graph coloring problem consists of a graph, a specified number of colors, and the requirement to find a color for each node in the graph such that no pair of adjacent nodes (i.e., nodes linked by an edge in the graph) have the same color. Graph coloring has received considerable attention and a number of search methods have been developed [Minton et al., 1990, Johnson et al., 1991, Selman et al., 1992]. This is a well-known NP-complete problem whose solution cost grows exponentially in the worst case as the size of the problem (i.e., number of nodes in the graph) increases.

For this problem, the average degree of the graph γ (i.e., the average number of edges coming from a node in the graph) is an order parameter that distinguishes relatively easy from harder problems, on average. In this paper, we focus on the case of 3–coloring (i.e., when 3 different colors are available), for which the transition between under- and overconstrained problems and hence the region of hardest problems occurs near [Cheeseman et al., 1991] $\gamma = 5$. While there are likely to be additional order parameters, such as the variance in the degrees, this one was sufficient to allow us to find a set of graphs that are relatively hard to color with 3 colors.

In our experiments we used two very different search methods. The first was a complete, depth-first backtracking search based on the Brelaz heuristic [Johnson et al., 1991] which assigns the most constrained nodes first (i.e., those with the most distinctly colored neighbors), breaking ties by choosing nodes with the most uncolored neighbors (with any remaining ties broken randomly). For each node, the smallest color consistent with the previous assignments is chosen first, with successive choices made when the search is forced to backtrack. This complete search method is guaranteed to eventually terminate and produce correct results. Moreover, it operates by attempting to extend partial colorings to complete solutions.

Our second method used heuristic repair [Minton et al., 1990] from randomly selected initial configurations. This method, which always operates with complete assignments (i.e., each node is assigned some color), attempts to produce a solution by selecting a node and changing its color to reduce as much as possible, or at least leave unchanged, the number of violated constraints in the problem. If some progress toward actually reducing the number of violations is not made within a prespecified number of steps, the search restarts from a new initial condition. This method is incomplete, i.e., if the problem has no solution it will never terminate. In practice, an upper bound on the total number of tries is made: if no solution is found, the method may incorrectly report there are no solutions.

For hard problems, both methods have a high variance in the number of steps required to find a solution. Moreover, their very different nature suggests a combination of the two methods will give a collection of agents far more diverse than if all agents use the same method. In particular, heuristic repair is often very effective at finding solutions once it starts "near" enough to one.

To generate hard problems we examined many random graphs with connectivity near the transition region. To correspond with the cooperative methods used previously [Clearwater et al., 1991] and simplify the use of hints, we considered only graphs that did in fact have solutions. Specifically, to construct our sample of graphs, we first divided the nodes into three classes (as nearly equal as possible) and allowed only edges that connected nodes in different classes to appear in our graph. This guaranteed that the graphs had a solution. Then trivial cases of underconstrained nodes were avoided by making sure each node had degree at least three. Finally, additional edges required to reach the desired connectivity were then added randomly. Many of the resulting graphs were trivial to search (e.g., for 100 node graphs, the median search cost for the Brelaz heuristic was about 200 steps at the peak). To identify those that were in fact difficult, the resulting graphs were searched repeatedly with both search methods, and only those with high search cost for all these trials were retained. This selection generally produced hard graphs with search costs one to three orders of magnitude higher than typical cases. We should also note that these graphs were hard even when compared to other methods of generating graphs which are known to give harder cases on average. Specifically, the prespecification of a solution state in our method tends to favor graphs with many solutions and hence favors easier graphs than uniform random selection. For 100 node graphs, this latter method gives a peak median search cost of about 350 steps. Even more difficult cases are emphasized by restricting consideration to graphs with no trivial reductions with typical costs of about 1000 steps [Cheeseman et al., 1991].

A Cooperative Search

There are two basic steps in implementing a cooperative search based on individual algorithms. First, the algorithms themselves must be modified to enable them to produce and incorporate information from other agents, i.e., read and write hints. Second, decisions as to exactly what information to use as hints, when to read them, etc. must be made. We should note that the first step may, in itself, change the performance of the initial algorithm or its characteristics (e.g., changing a complete search method into an incomplete one). Since this may change the absolute performance of the individual algorithm, a proper evaluation of the benefit of cooperation should compare the behavior of multiple agents, exchanging hints, to that of a single one running the same, modified, algorithm, but unable to communicate with other agents. In that way, the effect of cooperation, due to obtaining hints from other agents, will be highlighted.

For example, a single agent running the Brelaz algorithm can first be modified so that it may read and write hints (it itself produced) from a private blackboard. This alone leads to slightly improved performance. The effect of cooperation can then be assessed by comparing a society of N agents each running the modified algorithm in isolation with N agents running the same algorithm except that they read and write to a common blackboard. In this way we can subtract out the effects of the changed algorithm and the memory capacity on the performance of the agents, leaving just the effect of cooperation.

While there are many ways to use hints, we made fairly simple choices similar to those used previously [Clearwater et al., 1991], in which hints consisted of partial solutions (thus for graph coloring, these hints are consistent colorings for a subset of the nodes in the graph). A central blackboard, of limited size, was used to record hints produced by the agents. When the blackboard was full, the oldest (i.e., added to the blackboard before any others) of the smallest (i.e., involving colors for the fewest nodes) hints were overwritten with new hints.

Each agent independently writes a hint, based on its current state, at each step with a fixed probability q. When an agent was at an appropriate decision point, described below, it read a hint with probability p. Otherwise, or if there were no available hints, it continued with its original search method. Thus, setting p to zero corresponds to independent search since hints would never be used. We next describe how the two different search methods were modified to produce and incorporate hints.

Using hints with the Brelaz heuristic

At any point in a backtracking search, the current partial state is a consistent coloring of some subset of the graph's nodes. When writing a hint to the blackboard, the Brelaz agents simply wrote their current state.

Each time the agent was about to expand a node in its backtrack search, it would instead, with probability p, attempt to read a compatible hint from the blackboard, i.e., a hint on the blackboard whose assignments were 1) consistent with those of the agent (up to a permutation of the colors[1]) and 2) specified at least one node not already assigned in the agent's current state. Frequently, there was no such compatible hint (especially when the agent was deep in the tree and hence had already made assignments to many of the nodes), in which case the agent continued with its own search.

When a compatible hint was found, its overlap with the agent's current state was used to determine a permutation of the hint's colors that made it consistent with the state. This permutation was applied to the remaining colorings of the hint and then used to extend the agent's current state as far as possible (ordering the new nodes as determined by the Brelaz heuristic), and retaining necessary backtrack points so that the overall search remained complete. In effect, this hint simply replaced decisions that the Brelaz heuristic would have made regarding the initial colors to try for a number of nodes.

[1]We thus used the fact that, for graph coloring, any permutation of the color assignments for a consistent set of assignments is also consistent.

Using hints with heuristic repair

With heuristic repair, the agent's state always has a color assignment for each node, but it will not be fully consistent (until a solution is found). In order to produce a consistent partial assignment for use as a hint, we started with the full state and randomly removed assignments until a hint with no conflicts was obtained.

The heuristic repair agents have a natural point at which to read hints, namely when they are about to start over from a new initial state. At these times, we had each agent read a random hint from the blackboard with probability p, and otherwise randomly generate a new state. This hint, consisting of an assignment to some of the nodes, overwrote the agent's current state.

How Can Cooperation Help?

A simple explanation of the potential benefit of cooperative search is given by observing that the hints provide consistent colorings for large parts of the graph. Agents reading hints in effect then attempt to combine them with their current state. Although not always successful, those cases in which hints do combine well allow the agent to proceed to a solution by searching in a reduced space of possibilities. Even if many of the hints are not successful, this results in a larger variation of performance and hence can still improve the performance of the group when measured by the time for the first agent to finish.

As a more formal, but oversimplified, argument, suppose we view the agents as making a series of choices. Let p_{ij} be the probability that agent i makes choice j correctly (i.e., in the context of its previous choices, this one continues a path to a solution, e.g., by selecting a useful hint). The probability that the series of choices for agent i is correct is then just $p_i = \prod_j p_{ij}$. With sufficient diversity in the hints and agents' choices to prevent the p_{ij} from being too correlated, and viewing them as random variables, this multiplicative process results in a lognormal distribution [Redner, 1990] for agent performance, i.e., a random variable whose logarithm is normally distributed. This distribution has an extended tail compared to, say, a normal distribution or the distribution of the performance of the Brelaz heuristic on many graphs. Hence there is an increased likelihood that at least one agent will have much higher than average performance, leading to an improvement in group performance.

In practice, this simple argument must be modified to account for the possibility of backtracking and the fact that the quality of the hints changes during the search [Clearwater et al., 1992], but nevertheless gives some insight into the reason for the improvement and highlights the importance of maintaining diversity. Because of the high intrinsic variance in performance near the transition point, this in turn suggests why these cooperative methods are likely to be most applicable for the hard problems in the transition region.

Experimental Results

In this section we compare the behavior of independent searches with the cooperative method described above for some hard to color graphs. A simple performance criterion is the number of search steps required for the first agent to find a solution. However, this could be misleading when agents use different search methods whose individual steps have very different computational cost. It also ignores the additional overhead involved in selecting and incorporating hints. Here we present data based on actual execution time of an unoptimized serial implementation of the searches in which individual steps are multiplexed (i.e., each agent in the group takes a single step, with this procedure repeated until one agent finds a solution). The results are qualitatively similar to those based on counting the number of steps [Hogg and Williams, 1993], with the main differences being due to 1) a hint-exchange overhead which made individual cooperative steps about 5% slower than the corresponding independent ones, and 2) heuristic repair steps being about 2.4 times faster than Brelaz ones. This latter fact means that in a parallel implementation of a mixed group, the heuristic repair searches would actually complete relatively more search steps than when run serially. A parallel implementation would also face possible communication bottlenecks at the central blackboard though this is unlikely to be a major problem with the small blackboards considered here due to the relatively low reading rate and the possibility of caching multiple copies of the blackboard which are only slowly updated with new hints. Thus we can expect the cooperative agents to gain nearly the same speedup from parallel execution as the independent agents, i.e., a factor of 10 for our group size. While this must ultimately be addressed by comparing careful parallel implementations, the improvement in the execution time reported below, as well as the reduced number of search steps [Hogg and Williams, 1993], suggest the cooperative methods are likely to be beneficial for parallel solution of large, hard problems.

In Figs. 1 and 2, we compare the performance of groups of 10 cooperative agents with the same number of agents running independently. Note that in both cases, cooperation generally gives better performance than simply taking the best of 10 independent agents. Moreover, cooperation appears to be more beneficial as problem hardness (measured by the performance of a group of independent

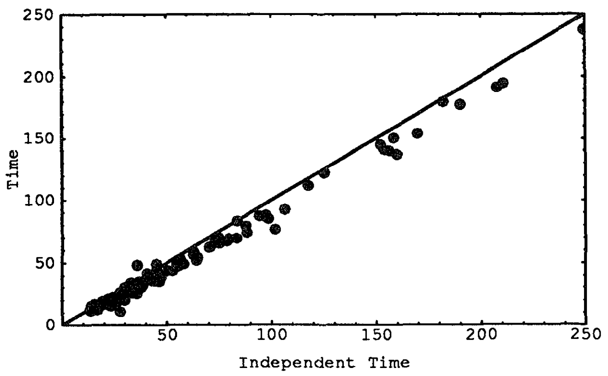

Fig. 1. Performance of groups of 10 cooperating agents vs. that of groups of 10 independent agents, using the Brelaz search method. Each point corresponds to a different graph and is the median, over 10 trials, of the execution time in seconds, on a SparcStation 2, required for the first agent in the group to find a solution. Each second of execution corresponds to about 90 search steps for each of the 10 agents. For comparison, the diagonal line shows the values at which cooperative and independent performance are equal. Cooperation is beneficial for points below this line. In these experiments, the blackboard was limited to hold 100 hints, and we used $p = 0.5$; $q = 0.1$ and graphs with 100 nodes.

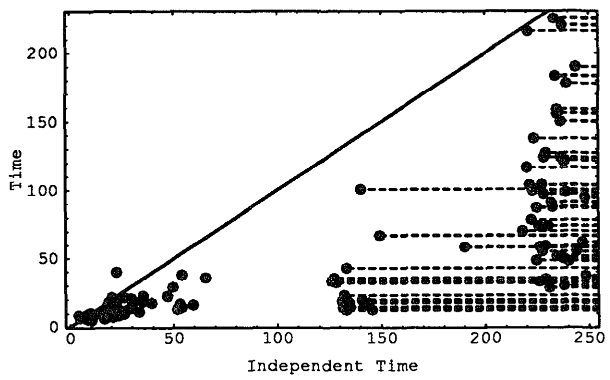

Fig. 2. Cooperation with heuristic repair. Each second of execution corresponds to about 210 search steps for each of the 10 agents. For comparison, the diagonal line shows the values at which cooperative and independent performance are equal. Some of the independent agent searches did not finish within 50000 steps at which point they were terminated. In these cases, the median performance shown in the plot for the independent agents is actually a lower bound: the dashed lines indicate the possible range of independent agent performances. Search parameters are as in Fig. 1.

agents) increases. We obtained a few graphs of significantly greater hardness than those shown in the figures which confirm this trend. We also observed that typically only a few percent of the hints on the blackboard were subsets of *any* solution so it is not obvious *a priori* that using these hints should be helpful at all.

Finally, Fig. 3 shows a combined society of agents. In this case, half the agents use the Brelaz method, half use heuristic repair, and hints are exchanged among all agents. Again, we see the benefit gained from cooperative search. For the graphs we generated, there was little correlation

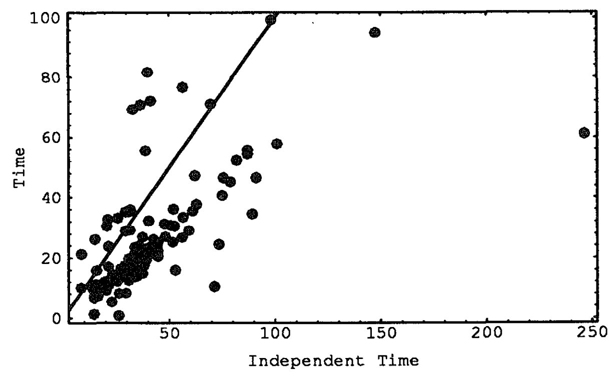

Fig. 3. Performance of groups of 10 cooperating agents, 5 using Brelaz and 5 using heuristic repair vs. the performance of the same groups searching independently. Each second of execution corresponds to about 120 search steps for each of the 10 agents. For comparison, the diagonal line shows the values at which cooperative and independent performance are equal. Search parameters are as in Fig. 1.

between solution cost for the two search methods, so that even when the agents were independent, this mixed society generally performed better than all agents using a single method.

While these results are encouraging, we should note that further work is needed to determine the best ways to exchange hints in societies using multiple methods, as well as the relative amount of resources devoted to different methods. Of particular interest is allowing the mix of agent types to change dynamically based on progress made during the search. More fundamentally for this avenue of research is understanding precisely what aspects of an ensemble of problems (e.g., in this case, determined by the precise method we used to generate the graphs) are important for the benefit of cooperation and the design of effective hints. Possibilities include the variance in individual search performance, the relative hardness of the graph and the proximity to the phase transition point.

Conclusion

In summary, we have tested our conjecture that cooperative methods are particularly well suited to hard graph coloring problems and seen that, even using simple hints, they can improve performance. It is further encouraging that the basic concepts used here, from the existence of regions of hard search problems characterized by order parameters to the use of partial solutions as hints, are applicable to a wide range of search problems.

There are a number of questions that remain to be addressed. An important one is how the observed cooperative improvement scales, both with problem size and, for a fixed size, with changes in the order parameters determining problem difficulty. There is also the question of how much different a parallel implementation is.

Another issue concerns whether these ideas can be applied to problems with no solutions to more quickly determine that fact. This is particularly relevant to the hard problems since they appear to occur at or near the transition from under- to overconstrained problems which have many and no solutions respectively. In cases with no solution, either one must compare complete search methods (e.g., by having at least one agent use a complete search method even when reading hints) or else evaluate both search speed and search accuracy to make a valid comparison. More generally, when applied to optimization problems, one would need to consider quality of solutions obtained as well as time required.

This study also raises a number of more general issues regarding the use of hints. As we have seen, diversity can arise from the intrinsic variation near the transition point and from random differences in the use of hints. Nevertheless, as more agents, using the same basic method, are added diversity does not increase as rapidly [Clearwater et al., 1992]. This suggests more active approaches to maintaining diversity, such as explicitly partitioning the search space or, more interestingly, combining agents using different search methods such as genetic algorithms [Goldberg, 1989] or simulated annealing [Johnson et al., 1991].

From a more practical point of view, a key decision for designing cooperative methods is how hints are generated and used, i.e., the "hint engineering". This involves a number of issues. The first is the nature of the information to exchange. This could consist of any useful information concerning regions of the search space to avoid or likely to contain solutions. The next major question is when during its search should an agent produce a hint. With backtracking, the agent always has a current partial solution which it could make available by placing it on a central blackboard. Generally, agents should tend to write hints that are likely to be useful in other parts of the search space. Possible methods to use include only writing the largest partial solutions an agent finds (i.e., at the point it is forced to backtrack) or only if the hint is at least comparable in size to those already on the blackboard. Complementary questions are when should an agent decide to read a hint from the blackboard, which one should it choose and how should it make use of the information for its subsequent search. Again there are a number of reasonable choices which have different benefits, in avoiding search, and costs for their evaluation, as well as more global consequences for the diversity of the agent population. For instance agents could select hints whenever a sufficiently good hint is available, or whenever the agent is about to make a random choice in its search method (i.e., use the hint to break ties), or whenever the agent is in some sense stuck, e.g., needing to backtrack or at a local optimum. For deciding which available hint to use, methods range from random selection [Clearwater et al., 1991] to picking one that is a good match, in some sense, to the agent's current state. Final issues are the hint memory requirements and what to discard from a full blackboard.

Given this range of choices, is there any guidance for making good use of hints? Theoretical results [Clearwater et al., 1992] emphasize the use of diversity for good cooperative performance. As a note of caution in developing more sophisticated hint strategies, the choices should promote high diversity among the agents [Huberman, 1990, Hogg, 1990] giving many opportunities to try hints in different contexts. This means that choices that appear reasonable when viewed from the perspective of a single agent, could result in lowered performance for the group as a whole, e.g., if all agents are designed to view the same hints as the best to use.

As with other heuristic techniques, the detailed implementation of appropriate choices to maintain diversity of the group of agents while also maintaining reasonable individual performance remains an empirical issue. While we can expect further improvements from more sophisticated use of hints, the fact our relatively simple mechanisms are able to give increased performance suggests that such methods may be quite easily applied.

Acknowledgments

We have benefited from discussions with S. Clearwater.

References

Cheeseman, P., Kanefsky, B., and Taylor, W. M. (1991). Where the really hard problems are. In Mylopoulos, J. and Reiter, R., editors, *Proceedings of IJCAI91*, pages 331–337, San Mateo, CA. Morgan Kaufmann.

Clearwater, S. H., Huberman, B. A., and Hogg, T. (1991). Cooperative solution of constraint satisfaction problems. *Science*, 254:1181–1183.

Clearwater, S. H., Huberman, B. A., and Hogg, T. (1992). Cooperative problem solving. In Huberman, B., editor, *Computation: The Micro and the Macro View*, pages 33–70. World Scientific, Singapore.

Fishburn, J. P. (1984). *Analysis of Speedup in Distributed Algorithms*. UMI Research Press, Ann Arbor, Michigan.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, NY.

Helmbold, D. P. and McDowell, C. E. (1989). Modeling speedup(n) greater than n. In Ris, F. and Kogge, P. M., editors, *Proc. of 1989 Intl. Conf. on Parallel Processing*, volume 3, pages 219–225, University Park, PA. Penn State Press.

Hogg, T. (1990). The dynamics of complex computational systems. In Zurek, W., editor, *Complexity, Entropy and the Physics of Information*, volume VIII of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 207–222. Addison-Wesley, Reading, MA.

Hogg, T. and Williams, C. P. (1993). Solving the really hard problems with cooperative search. In Hirsh, H. et al., editors, *AAAI Spring Symposium on AI and NP-Hard Problems*, pages 78–84. AAAI.

Huberman, B. A. (1990). The performance of cooperative processes. *Physica D*, 42:38–47.

Imai, M., Yoshida, Y., and Fukumura, T. (1979). A parallel searching scheme for multiprocessor systems and its application to combinatorial problems. In *Proc. of IJCAI-79*, pages 416–418.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. (1991). Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning. *Operations Research*, 39(3):378–406.

Kornfeld, W. A. (1981). The use of parallelism to implement a heuristic search. In *Proc. of IJCAI-81*, pages 575–580.

Mehrotra, R. and Gehringer, E. F. (1985). Superlinear speedup through randomized algorithms. In Degroot, D., editor, *Proc. of 1985 Intl. Conf. on Parallel Processing*, pages 291–300, Washington, DC. IEEE.

Minton, S., Johnston, M. D., Philips, A. B., and Laird, P. (1990). Solving large-scale constraint satisfaction and scheduling problems using a heursitic repair method. In *Proceedings of AAAI-90*, pages 17–24, Menlo Park, CA. AAAI Press.

Mitchell, D., Selman, B., and Levesque, H. (1992). Hard and easy distributions of SAT problems. In *Proc. of 10th Natl. Conf. on Artificial Intelligence (AAAI92)*, pages 459–465, Menlo Park. AAAI Press.

Pramanick, I. and Kuhl, J. G. (1991). Study of an inherently parallel heuristic technique. In *Proc. of 1991 Intl. Conf. on Parallel Processing*, volume 3, pages 95–99.

Rao, V. N. and Kumer, V. (1992). On the efficiency of parallel backtracking. *IEEE Trans. on Parallel and Distributed Computing*.

Redner, S. (1990). Random multiplicative processes: An elementary tutorial. *Am. J. Phys.*, 58(3):267–273.

Selman, B., Levesque, H., and Mitchell, D. (1992). A new method for solving hard satisfiability problems. In *Proc. of 10th Natl. Conf. on Artificial Intelligence (AAAI92)*, pages 440–446, Menlo Park, CA. AAAI Press.

Williams, C. P. and Hogg, T. (1992a). Exploiting the deep structure of constraint problems. Technical Report SSL92-24, Xerox PARC, Palo Alto, CA.

Williams, C. P. and Hogg, T. (1992b). Using deep structure to locate hard problems. In *Proc. of 10th Natl. Conf. on Artificial Intelligence (AAAI92)*, pages 472–477, Menlo Park, CA. AAAI Press.