Systematic Versus Stochastic Constraint Satisfaction

Eugene C. Freuder*(Chair)
Department of Computer Science
University of New Hampshire
Durham, NH 03824 U.S.A.
ecf@cs.unh.edu

Rina Dechter†
Deptartment of Information and Computer Science
University of California, Irvine
Irvine, CA 92717 U.S.A.
dechter@ics.uci.edu

Matthew L. Ginsberg‡
CIRL
University of Oregon
Eugene, Oregon 97403 U.S.A.
ginsberg@cs.uoregon.edu

Bart Selman
AT&T Bell Laboratories
600 Mountain Avenue, Room 2T-414
Murray Hill, NJ 07974 U.S.A.
selman@research.att.com

Edward Tsang
Department of Computer Science
University of Essex
Essex CO4 3SQ, United Kingdom
edward@essex.ac.uk

Abstract

This panel explores issues of systematic and stochastic control in the context of constraint satisfaction.

1 Introduction: Eugene C. Freuder

Constraint satisfaction problems (CSPs) involve finding values for problem variables that satisfy restrictions on which combinations of values are allowed [Freuder and Mackworth, 1994; Tsang, 1993]. They have many applications, including planning and scheduling, design and configuration, vision and language, temporal and spatial reasoning. The map coloring problem is a simple example, where the problem variables correspond to countries, the values to colors, and the constraints specify that neighboring countries cannot have the same color.

In constraint satisfaction, as in many areas of AI, there is competition between systematic methods for problem solving, which will, if necessary, explore the entire problem space, and stochastic methods, which can involve an element of random choice as to which portions of the problem space will be examined. Systematic CSP aproaches include search and inference techniques. Stochastic approaches include those influenced by physical metaphors (hill climbing, simulated annealing) and those influenced by biological metaphors (neural networks, genetic algorithms). Each of our panelists is prepared to speak for one of these four approaches (but will not be restricted to that limited advocacy role).

Though our primary focus is on systematic versus stochastic, there are also interesting issues of search versus inference on the systematic side, and physical versus biological on the stochastic side. On the other hand, there are intriguing possibilities for conjoining the systematic and the stochastic.

Stochastic methods are generally incomplete: they cannot guarantee that they will find a solution, or prove that a solution does not exist. Systematic methods can. However, stochastic methods are sometimes very fast. Are stochastic methods inherently faster? Or are there some types of problems for which stochastic methods are better suited, and some types of problems for which systematic methods are better suited?

If and when stochastic methods are faster, why are they faster? There are actually at least three factors that these methods generally have in common – randomness, locality, incompleteness. Which are really operative factors?

Is the random element important? Why? There is often a local nature to stochastic search, decision making based only on local features of the problem space: how can making less informed decisions be helpful? Is there

*This material is based on work supported by the National Science Foundation under Grant No. IRI-9207633.

†This work was partially supported by NSF grant IRI-9157636, by the Electrical Power Research Institute (EPRI), and by grants from Toshiba of America, Xerox Northrop and Rockwell.

‡This work has been supported by the Air Force Office of Scientific Research under contract 92-0693 and by ARPA/Rome Labs under contracts F30602-91-C-0036 and F30602-93-C-00031.

actually a positive advantage to randomness or locality; or is it simply unproductive to spend more time to make informed or global choices? On the surface it seems plausible that one could gain something – speed – by giving up something – completeness: but can we explain how this tradeoff works?

Do the successes of stochastic methods really depend on their being non-systematic, or can they be incorporated into systematic methods? For example, an early success with constraint satisfaction using neural networks, led to a hill climbing technique, that led to ordering heuristics for systematic backtrack search. The systematic version did at least as well as the hill climbing in at least one of the tested problem domains [Minton *et al.*, 1992].

Stochastic methods often begin by transforming a problem into a more "atomistic" representation, e.g. in terms of SAT propositional variables or neural network nodes. These representations would appear to lose, or at least deemphasize, higher level organizational information; are they succeeding in spite of that, or because of it? Are these methods succeeding because they emphasize the lower level "microstructure" of problems, the consistency patterns as opposed to the higher level patterns of constraints or subproblems [Freuder, 1994]?

Some additional provocative questions for the stochastic side:

- Time versus Space. Are stochastic methods buying speed with unreasonable space demands? To what extent do these methods only appear to solve large problems because they have a large representation of a small problem?

- Learning versus Cheating. Are stochastic methods being fairly tuned for a given problem domain, or unfairly biased?

- Consistency Computation versus Control Computation. How would stochastic methods fare on problems where there is a high cost associated with determining consistency of individual values, enhancing the effectiveness of inference that avoids consistency checking?

Some additional provocative questions for the systematic side:

- Completeness versus Responsiveness. To what exent is completeness a red herring: if we cannot wait for the complete answer, is not the systematic method de facto incomplete?

- Intelligence versus Brute Force. Does systematic reasoning pay off or is intelligence just overhead?

- Optimal versus Satisficing. Will repeated application of stochastic methods find a "good enough" solution faster than a systematic search for the best solution?

Ultimately the most provocative question for both sides may be:

- Rather than competing, can we cooperate?

Can the two communities learn from each other? Which differences are important and which superficial, or even misleading? Can methods usefully combine ideas from both communities? There has been exciting recent progress in that direction ([Ginsberg and McAllester, 1994; Verfaillie and Schiex, 1994; Yokoo, 1994]).

2 Systematic vs Stochastic Greedy Algorithms: Rina Dechter

Systematic algorithms have two properties [Pearl, 1984]: (1) Do not leave any stone unturned (completeness), and (2) do not turn any stone more then once (efficiency). Most classical research in heuristic search focused on such algorithms, some examples of which are A^* and backtracking. Recently, greater attention is shifting towards a class of greedy nonsystematic algorithms that we will call *Stochastic greedy (SG)*. These algorithms may leave "many stones unturned" and may also "turn the same stone multiple times" (as do many complete algorithms).

Here are some facts: first, it can be demonstrated that many large problem instances that are practically unsolvable by systematic algorithms are solved efficiently using stochastic-greedy methods [Minton *et al.*, 1992; Selman *et al.*, 1992]. Secondly, it was also observed that for many problem instances SG methods are less effective than systematic search, even when the problem is satisfiable [Konolige, 1994; Kask & Dechter, 1995]. Third, SG methods are highly ineffective for inconsistent problems. Finally, constraint satisfaction problems are NP-complete and therefore finding one "best performing" algorithm on every problem instance is unlikely.

The important problem is not finding which paradigm is the winner, but rather how to exploit identified *class-superior* algorithms within one meta framework. For argument's sake, let us call an algorithm "class-superior" if there exists a reasonable-size class of problem instances over which the algorithm is proven superior, be it experimentally or theoretically. Let us also assume that we have identified a collection of classes and their corresponding class-superior algorithms and that no algorithm dominates other on all classes. I make two claims: (1) that this hypothetical picture is where we stand today, and (2) that it makes no sense given such a picture to talk about a winner. Instead it makes more sense to talk about "integration" of algorithms among those identified as class-superior.

SG algorithms should surely be added to the arsenal of *class-superior* algorithms, as should incomplete and polynomial consistency enforcing (if we consider the immense class of unsatisfiable problems) [Mackworth & Freuder, 1984].

A brute-force approach of combining *class-superior* algorithms would be by *parallel integration*. Namely, to run all algorithms in parallel and stop with the first one to finish [Hogg & Williams, 1994](*sequential integration* is another brute-force option where algorithms are run in sequence, each being stopped after a specified amount of time). A second approach would be to understand what problem features make a given algorithm run more effectively (e.g., identify tractable classes) and to exploit those features in the combining rule yielding a *case-based integration*. Lastly, algorithms that address different and orthogonal aspects of the search (e.g., look-back vs look-ahead enhancements to backtracking) can be integrated into one umbrella algorithm (*unconditional integration*) which may improve on each individual participating algorithm over the union of their preferred instances. Such an integrated algorithm can replace its individual components in the arsenal of class-superior algorithms for further case-based or parallel-based integration (e.g, backjumping integrated with Dynamic variable ordering (DVO), outperforms both) [Frost and Dechter, 1994]. Another proposed integration is dynamic backtracking that integrates backjumping with some look-ahead ideas [Ginsberg, 1995].

Research in constraint networks in the last decade has lead to substantial understanding resulting in a list of class-superior algorithms, including backtracking-style algorithms like backjumping, backmarking, constraint learning, forward-checking, dynamic variable ordering; structure exploiting algorithms like adaptive-consistency, tree-clustering, and cycle-cutset decomposition; as well as consistency enforcing algorithms like arc and path consistency algorithms [Dechter 1990].

To be able to exploit SG algorithms within an integrated algorithm, similar understanding should be acquired. Here are some sample questions: Are there polynomially recognizable conditions under which SG methods are guaranteed to provide a solution and in polynomial time? Can some form of preprocessing help SG methods? Can some form of learning during search help? Is there a set of enhancement schemes which are unanimously desirable?

In our efforts to answer some of these questions we focused on trying to improve SG performance on problems with sparse structure using two approaches: by using consistency enforcing algorithms prior to SG [Kask & Dechter, 1995] and by adapting the SG algorithm to be sensitive to the problem's structure [Pinkas and Dechter, 1995].

A note on "local vs global" search. SG methods are often called local while systematic methods are associated with "global" search. The notion of "locality" is dependent on the selected search space definition with its associated neighborhood relation. What is local in one space is global in the other. Backtracking is a local search procedure that can be viewed as selecting the next state with the maximum gradient increase of some look-ahead function.

3 Systematic and Nonsystematic Search: Matthew L. Ginsberg

There seems to be a great deal of confusion here regarding the question of whether or not systematic search methods can be applied to problems of interesting size. Tsang, for example, says:

> Complete algorithms alone have little chance of solving realistic CSPs [constraint satisfaction problems]. For example, the problem of assigning 30 jobs to 10 machines (satisfying certain constraints) has a search space of 10^{30} nodes.

So what? Neither a systematic nor a nonsystematic approach needs to search the entire space in order to solve a satisfiable problem. Perhaps what Tsang is trying to say is simply that there is no point in trying to search enormous search spaces in their entirety. Assuming that $P \neq NP$, it's hard to argue with that.

The upshot of this observation seems to be that if a large problem is unsatisfiable, finding a proof of unsatisfiability is likely to be impossible for *any* approach. One can conclude from this that the principal claimed advantage of systematic methods, that they can eventually report a problem as unsolvable, is illusory.

Of course, that's simply not true. There are many large problems that are clearly unsatisfiable. Can United Airlines schedule all of its daily flights using only 17 aircraft? Surely not – and we don't need to examine the entire space of possible schedules in order to prove it. So we can probably say the following:

Observation 1 *Some – but not all – large unsatisfiable problems are unlikely to be solvable using systematic search methods. All unsatisfiable problems of any size are unsolvable using nonsystematic methods.*

Contrary to the nonsystematic community's propaganda, unsatisfiable problems are important, and can sometimes – not always – be solved using systematic methods.

The claimed advantage of the nonsystematic methods comes from their superior performance on *satisfiable* problems. There has been a great deal written and said about this [Langley, 1992; Minton *et al.*, 1992; Selman *et al.*, 1992], and I won't repeat it here. A reasonable summary appears to be that the symbolic approaches (min-conflicts, GSAT and so on) uniformly outperform nonsymbolic methods (simulated annealing, neural nets, etc.) and that:

Observation 2 *When solving a satisfiable problem, it is important to be able to follow local gradients.*

All of the claimed performance advantage of the stochastic or other nonsystematic methods appears to be based in their ability to move locally through the search space; systematic methods typically lack this property.[1] But nowhere is it suggested that nonsystematicity in and of itself is an attractive property; how could it be?

In spite of their inability to follow local gradients, systematic approaches are competitive with nonsystematic ones on a wide variety of problems; job-shop scheduling is a typical example [Harvey and Ginsberg, 1995; Smith and Cheng, 1993; Smith, 1992]. One explanation for the performance of the systematic approach is that some measure of systematicity is needed to escape the local minima that arise in realistic problems.

When solving a mix of problems including both satisfiable and unsatisfiable instances, it is clear that systematicity and the ability to follow local gradients are both important. Given the job-shop scheduling results, it appears that both can be important even if we restrict our attention to satisfiable problems alone.

One member of the panel (Dechter), reaching the same conclusion, suggests that the algorithms run in parallel or that we strive to "understand what problem features make a given algorithm run more effectively ... yielding a case-based integration." She suggests that algorithm development should perhaps be regarded as less important than this integration, saying that, "The important problem is ... how to exploit identified class-superior algorithms within one meta-framework."

This seems to me to miss the mark completely. Far more sensible would be for the constraint-satisfaction community to focus at least some of its effort on the development of new systematic techniques that follow local gradients. Both dynamic backtracking [Ginsberg, 1993] (misdescribed by Dechter as an integration of backjumping and lookahead) and partial-order dynamic backtracking [Ginsberg and McAllester, 1994] are preliminary attempts in this direction. The work is much more difficult than simple and continued experimentation on ever-larger randomly generated problems, but seems far more likely to lead to results of practical importance.

4 Stochastic Search For Model Finding: Bart Selman

Recent work on stochastic search for solving constraint satisfaction problems has shown that that such methods can be surprisingly good at finding *globally* optimal solutions. For example, in our work on Boolean satisfiability testing we have found that, when given a set of propositional clauses, a greedy local search method can often find a truth assignment that satisfies all clauses in the input set [Selman *et al.* 1992]. This came as a surprise to us, since we assumed that local search methods would tend to get stuck in local optima, and thus might find good *near-optimal* solutions, but not completely satisfying assignments. In dealing with combinatorial optimization problems, finding good near-optimal solutions is quite useful, but in many AI applications, near-satisfying assignments are of little value. For example, in satisfiability encodings of planning problems, a near-satisfying assignment corresponds to a plan with a "magic" step, i.e., a physically infeasible operation. So, the fact that stochastic search methods often find completely satisfying solutions significantly widens the range of potential applications of such search methods. (See also [Minton *et al.* 1990].)

One obvious drawback is that stochastic search cannot be used to prove the *inconsistency* of a set of constraints. This is a problem in applications based on theorem proving. One way to show that a formula α follows from a theory Σ, is to show that Σ with the negation of α is inconsistent. Unfortunately, since stochastic search methods are incomplete, they cannot be used to prove inconsistency, simply because these methods do not systematically explore all possible models. Therefore, in order to take full advantage of stochastic search methods, it is necessary to formulate tasks not in terms of theorem proving, but rather in terms of model finding.

It has recently been shown that systematic methods can be improved by taking advantage of principles behind stochastic (or local) search. This has led some people to suggest that perhaps in the end we will be able to develop systematic methods that will be as good as stochastic search methods at showing the consistency of a set of constraints, but, in addition, can also discover inconsistencies. As a consequence, there may not be much practical difference between model-finding and theorem proving. I will argue that this is unlikely. More specifically, I will discuss the fundamental asymmetry between the task of showing a set of constraints to be consistent versus showing them to be inconsistent. Of course, this issue is closely related to the traditional distinction between NP and co-NP. I will argue that this is not just an interesting theoretical distinction but that there are good reasons to believe that this distinction has concrete practical consequences. The difference has been ignored in the past because of the traditional emphasis on systematic search methods, for which the distinction between NP and co-NP has no practical consequences. The recent work on stochastic search, however, suggests that there may in fact be a real difference in practical terms. Consider, for example, randomly generated satisfiability problems. Stochastic search methods can be used to find satisfying assignments of hard random 3SAT formulas with up to 2500 variables. However, the current best systematic methods can show the inconsistency

[1] Dechter's suggestion in this panel that "Backtracking is a local search procedure that can be viewed as selecting the ... maximum gradient increase of some function" misses the point. We want to follow local gradients of a function that measures solution quality, not one that happens to mimic our particular search procedure.

of formulas with only up to about 400 variables [Crawford and Auton, 1993]. Moreover, a theoretical analysis of resolution proofs has shown that the *shortest* resolution proofs of most hard inconsistent instances are of exponential size [Chvatal and Szemeredi, 1988]. Since backtrack-style search can often be viewed as a variant of resolution, the exponential lower-bound on resolution places a hard limit on what one can expect from most backtrack-style procedures. What is needed to tackle larger formulas are more powerful proof systems, such as extended resolution, where one allows new variables to be introduced in the proof. Such systems might give us shorter proofs of the inconsistency of certain large instances; unfortunately, we do not yet know whether shorter proofs actually exist for, *e.g.*, the hard random problems, nor do we know how to to search automatically for such shorter proofs.

In summary, the recent work on stochastic search methods has revealed an interesting asymmetry: it is generally easier to find a satisfying assignment of a consistent set of constraints, than to show a set of constraints to be inconsistent. This asymmetry suggests that it is preferable to formulate problems in terms of model-finding instead of as theorem proving tasks. A tantalizing possibility is that in order to handle certain larger inconsistent sets of constraints, we may have to develop stochastic methods that search for short proofs expressed in more powerful proof systems.

5 Case for Stochastic Search: Edward Tsang

This debate is important because research direction is often more important than the development of techniques. In the past the majority of research in constraint satisfaction has focussed on complete search algorithms and heuristics to be employed by them [Tsang, 1993]. Given the NP-complete nature of constraint satisfaction problems (CSPs), should complete algorithms be given so much attention and incomplete search algorithms so little? What should their roles in constraint satisfaction applications be?

The advantage of tackling problems with complete algorithms is obvious. However, it is also important to realize their limitations. The fact is, complete algorithms alone have little chance of solving realistic CSPs. For example, the problem of assigning 30 jobs to 10 machines (satisfying certain constraints) has a search space of 10^{30} nodes. Even if we generously assume that a very, very efficient complete search algorithm only needs to look at one in every 10^{10} nodes in the search space, it still needs to examine 10^{20} nodes. If a machine can examine 10^{10} nodes per second (which would be incredible, and require the compatibility checks to be very inexpensive computationally), then given that roughly 3×10^7 seconds are available in a year, this innocent looking 30-variables-10-values-each problem would require about 300 years to solve! Developing interactive systems to solve such problems by complete methods is out of the question.

Complete algorithms have other limitations. In some applications, one prefers to find optimal or near optimal solutions. Complete algorithms (such as bound and bound) need to explore the whole search space to establish that the solution found is optimal, and this is limited by the combinatorial explosion problem (explained above). Moreover, when a problem has no solution, most complete algorithms developed so far can only report that no solution exists. What one may need is a near-solution to guide one through constraints relaxation.

All this does not mean that complete algorithms are completely useless, but that they are very limited in their applications. They could be used for small problems. When a problem is decomposable, they could be used to solve some of the subproblems. In any case, one should always evaluate the possibility of complete algorithms being able to solve the given problem within the time available before looking at other methods.

Stochastic methods that have been looked at for constraint satisfaction include hill climbing (HC) [Minton *et al.*, 1992], connectionist methods [Davenport *et al.*, 1994], genetic algorithms (GAs) [Warwick and Tsang, 1994] and simulated annealing (SA) [Ghedira, 1994; Davis, 1987].

No one should believe that the networks that people develop in connectionist methods realistically resemble the brain. Neither does any GA model evolution closely. But nature often gives us invaluable inspiration, and these methods have already demonstrated their effectiveness in various domains, including constraint satisfaction.

There are many reasons why the above mentioned stochastic methods, which sacrifice completeness for efficiency, are practical answers to real life CSPs. One important advantage of these methods is that they can be designed to terminate within a specified time period. Besides, domain knowledge can be incorporated into these methods—for example, knowledge can be used in the formulation of a connectionist network and in the representation to be used by a GA.

Another important issue is that the above mentioned stochastic methods have great potential to be extended to tackle constraint optimization problems and partial constraint satisfaction problems. For example, penalties for violating individual constraints and costs for assigning certain values to individual variables can be incorporated into the weights of the connections and the fitness functions of a GA. In contrast, the role of problem reduction techniques in constraint optimization problems is far from clear, and the usefulness of most of the search strategies, including lookahead and learning, are doubtful in optimization problems.

One important criteria for evaluating how promising a technique is lies in the current development of hardware.

(Arguably, advances in hardware has been much more significant than advances in software in the past decade or two.) Parallel architectures have become more and more readily available. Previous research and the above example have shown that a polynomial number of processors is not going to be sufficient to contain the combinatorial explosion problem in complete search methods. On the other hand, connectionism by nature can easily take advantage of the availability of parallel architectures. GAs have also been shown to be able to exploit parallelism effectively. (e.g. see [Muhlenbein *et al.*, 1991]).

Reliability is often the greatest concerned when using stochastic methods. Indeed, some stochastic methods can easily be trapped in local optima, but connectionism and GAs can be very reliable. For example, [Davenport *et al.*, 1994] showed that GENET has a remarkably high success rate in solving CSPs (in fact, it has never missed a solution in binary CSPs). [Pinkas and Dechter, 1995 to appear] has shows that connectionist methods can be complete for certain types of problems. Quite a lot of success has been found by GAs in finding near-optimal solutions.

Real life constraint satisfaction applications often involve large numbers of variables and large domains. One is often given limited time to solve such problems; e.g. a schedule may need to be produced every night. Some applications demand interactive systems to be produced. Limited by the combinatorial explosion problem, complete algorithms are unlikely to be able to meet the requirements of these applications.

Some applications require the optimal solution or near-optimal solutions to be found. In other applications, when the problem is insoluble, near-solutions may be preferred to a report suggesting that the problem is insoluble. Complete algorithms have little to offer to these requirements.

Stochastic methods, especially connectionism and GAs, have had a lot of success in real life applications, so they should receive much more attention by constraint satisfaction researchers than they have so far. The questions that should interest us are: which stochastic method to use when? How reliable is a particular stochastic method? How many times does a particular stochastic method miss solutions when they exist? How successful is a particular stochastic method in finding near-optimal solutions (how close are they to the optimal)? These (rather than complete search algorithms) should be the focus of future constraint satisfaction research.

To summarize, complete search algorithms suffer from the combinatorial explosion problem in general and therefore cannot be expected to solve most real life problems. In order to bring constraint satisfaction research out of the laboratory and put it into applications, attention should be shifted to stochastic search in constraint satisfaction in the future.

References

[Chvatal and Meiri, 1988] V. Chvatal and E. Szemeredi. Many hard examples for resolution. *JACM*, vol. 35, no. 4, 1988, 759–208.

[Crawford and Auton, 1993] J.M. Crawford and L.D. Auton. Experimental results on the cross-over point in satisfiability problems. *AAAI93*, 1993.

[Davenport *et al.*, 1994] A. J. Davenport, E. P. K. Tsang, C. J. Wang, and K. Zhu. GENET: A connectionist architecture for solving constraint satisfaction problems by iterative improvement. In *Proc, 12th National Conference on Artificial Intelligence (AAAI)*, volume 1, pages 325–330, 1994.

[Davis, 1987] L. Davis. *Genetic algorithms and simulated annealing*. Research notes in AI. Pitman/Morgan Kaufmann, 1987.

[Dechter 1990] R. Dechter. "Constraint networks." *Encyclopedia of Artificial Intelligence* (2nd Ed.), John Wiley, New York, pp. 276-285, 1991.

[Freuder and Mackworth, 1994] E. Freuder and A. Mackworth, editors. *Constraint-Based Reasoning*. MIT Press, Cambridge, MA, USA, 1994.

[Freuder, 1994] E. Freuder. Exploiting structure in constraint satisfaction problems. In *Constraint Programming*, Mayoh, Tyugu, Penjam, editors, Springer-Verlag, Berlin, pp. 51-74, 1994.

[Frost and Dechter, 1994] D. Frost and R. Dechter. "In search of the best backtracking search" In *AAAI-94*, pp. 301–306, 1994.

[Ghedira, 1994] K. Ghedira. Dynamic partial constraint satisfaction by multi-agent and simulated annealing. In *Workshop on Constraint Satisfaction Issues Raised by Practical Applications, 11th European Conference on Artificial Intelligence*, 1994.

[Ginsberg and McAllester, 1994] Matthew L. Ginsberg and David A. McAllester. GSAT and dynamic backtracking. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Bonn, Germany, 1994.

[Ginsberg, 1993] Matthew L. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46, 1993.

[Hogg & Williams, 1994] T. Hogg and C. P. Williams. "Expected gains from parallelizing constraint solving for hard problems," in *AAAI-94*, pp. 331-336, 1994.

[Harvey and Ginsberg, 1995] William D. Harvey and Matthew L. Ginsberg. Limited discrepancy search. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995.

[Kask & Dechter, 1995] K. Kask and R. Dechter. "GSAT and local consistency," in *IJCAI-95*, 1995.

[Konolige, 1994] K. Konolige. "Easy to hard: Difficult problems for greedy algorithms" *Knowledge Representation*, pages 374-378, 1994.

[Langley, 1992] Pat Langley. Systematic and nonsystematic search strategies. In *Artificial Intelligence Planning Systems: Proceedings of the First International Conference*, pages 145–152. Morgan Kaufmann, 1992.

[Mackworth & Freuder, 1984] A. Mackworth and E. Freuder. "The complexity of some polynomial search algorithms for constraint satisfaction problems," *Artificial Intelligence*, Vol. 25, No. 1, 1984.

[Minton *et al.*, 1992] S. Minton, M. Johnston, A.B. Philips, and P. Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[Muhlenbein *et al.*, 1991] H. Muhlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17:619–632, 1991.

[Pearl, 1984] J. Pearl. "Heuristics, Intelligent search strategies for computer problem solving," Addison Wesley, 1984.

[Pinkas and Dechter, 1995 to appear] G. Pinkas and R. Dechter. On improving connectionist energy minimization. *Journal of Artificial Intelligence Research*, 1995, to appear.

[Selman *et al.*, 1992] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446, 1992.

[Smith and Cheng, 1993] Stephen F. Smith and Cheng-Chung Cheng. Slack-based heuristics for constraint satisfaction scheduling. *AAAI-93*, pp. 139–144, 1993.

[Smith, 1992] Douglas R. Smith. Transformational approach to scheduling. Technical Report KES.U.92.2, Kestrel Institute, 1992.

[Tsang, 1993] E.P.K. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1993.

[Warwick and Tsang, 1994] T. Warwick and E. P. K. Tsang. Using a genetic algorithm algorithm to tackle the processors configuration problem. In *Proc. ACM Symposium on Applied Computing (SAC)*, pages 217–221, 1994.

[Verfaillie and Schiex, 1994] G. Verfaillie and T. Schiex. Solution reuse in dynamic constraint satisfaction problems. *AAAI-94*, pp. 307–312, 1994.

[Yokoo, 1994] M. Yokoo. Weak-commitment search for solving constraint satisfaction problems. *AAAI-94*, pp. 313–318, 1994.