From: AAAI-94 Proceedings. Copyright © 1994, AAAI (www.aaai.org). All rights reserved.

Weak-commitment Search for Solving Constraint Satisfaction Problems

Makoto Yokoo
NTT Communication Science Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan
yokoo@cslab.kecl.ntt.jp

Abstract

The min-conflict heuristic (Minton *et al.* 1992) has been introduced into backtracking algorithms and iterative improvement algorithms as a powerful heuristic for solving constraint satisfaction problems. Backtracking algorithms become inefficient when a bad partial solution is constructed, since an exhaustive search is required for revising the bad decision. On the other hand, iterative improvement algorithms do not construct a consistent partial solution and can revise a bad decision without exhaustive search. However, most of the powerful heuristics obtained through the long history of constraint satisfaction studies (e.g., forward checking (Haralick & Elliot 1980)) presuppose the existence of a consistent partial solution. Therefore, these heuristics can not be applied to iterative improvement algorithms. Furthermore, these algorithms are not theoretically complete.

In this paper, a new algorithm called *weak-commitment* search which utilizes the min-conflict heuristic is developed. This algorithm removes the drawbacks of backtracking algorithms and iterative improvement algorithms, i.e., the algorithm can revise bad decisions without exhaustive search, the completeness of the algorithm is guaranteed, and various heuristics can be introduced since a consistent partial solution is constructed. The experimental results on various example problems show that this algorithm is 3 to 10 times more efficient than other algorithms.

Introduction

A Constraint Satisfaction Problem (CSP) is a general framework that can formalize various problems in AI, and many theoretical and experimental studies have been performed (Mackworth 1992). Recently, the min-conflict heuristic (Minton *et al.* 1992) has been identified as a powerful heuristic for finding one solution of a CSP. This heuristic can be described as follows: when deciding a variable value, it chooses the value that minimizes the number of constraint violations between other variables. This heuristic has been applied to backtracking algorithms (Minton *et al.* 1992) and iterative improvement algorithms (Minton *et al.* 1992; Morris 1993).

In backtracking algorithms, a consistent partial solution is constructed for a subset of variables, and this partial solution is extended by adding variables one by one until a complete solution is found. In the backtracking algorithm that incorporates the min-conflict heuristic (min-conflict backtracking), all variables are given tentative initial values. When a variable is added to the partial solution, its tentative initial value is revised so that the new value satisfies all constraints between the partial solution, and satisfies as many constraints between variables that are not included in the partial solution as possible.

The drawback of backtracking algorithms is as follows.

- **One mistake in the value selection is fatal.** In min-conflict backtracking, the partial solution constructed during the search process will not be revised unless it is proven that there exists no complete solution subsuming the partial solution. If the algorithm makes a bad selection of a variable value, the algorithm must perform an exhaustive search for the partial solution in order to revise the bad decision. When the problem becomes very large, doing such an exhaustive search is virtually impossible.

On the other hand, iterative improvement algorithms (Minton *et al.* 1992; Morris 1993; Selman, Levesque, & Mitchell 1992) do not construct a consistent partial solution. In these algorithms, a *flawed* solution containing some constraint violations is revised by local changes until all constraints are satisfied. The min-conflict heuristic is used as the basis for the local changes. In these algorithms, the value of one variable can be changed repeatedly without any exhaustive search. Therefore, one mistake in the value selection is not fatal and can be revised easily. However, the iterative improvement algorithms have the following drawbacks.

- **The completeness of the algorithms can not be guaranteed.** We say that an algorithm is complete if the algorithm is guaranteed to find one solution eventually when solutions exist; and when there exists no solution, the algorithm is guaranteed to find out the fact that there exists no solution and

terminate. One exception is the fill algorithm (Morris 1993), which is guaranteed to find a solution if there exists one. However, this algorithm is far less efficient than a similar incomplete algorithm called the breakout algorithm (Morris 1993). Also, the fill algorithm will not terminate when there exists no solution.

- **Introducing other heuristics is difficult.** The completeness of the algorithms may have only theoretical importance when solving large-scale problems. A more practical drawback is that we can not apply most of the powerful heuristics obtained through the long history of constraint satisfaction studies (e.g., forward checking (Haralick & Elliot 1980)) to iterative improvement algorithms, since these heuristics presuppose the existence of a consistent partial solution.

In this paper, a new algorithm called *weak-commitment* search which utilizes the min-conflict heuristic is developed. In this algorithm, all variables are given tentative initial values, and variables are added one by one to the consistent partial solution as in the min-conflict backtracking. This algorithm constructs a consistent partial solution, but commits to the partial solution *weakly*, in contrast to backtracking algorithms which never abandon a partial solution unless it turns out to be hopeless. Namely, this algorithm commits to the partial solution as long as it can be extended. However, when there exists no value for one variable that satisfies all constraints between the partial solution, this algorithm abandons the whole partial solution, and starts constructing a new partial solution from scratch, using the current value assignment as new tentative initial values.

This algorithm removes the drawbacks of backtracking algorithms and iterative improvement algorithms, i.e., the algorithm can revise bad decisions without exhaustive search, the completeness of the algorithm is guaranteed, and various heuristics can be introduced since a consistent partial solution is constructed.

In the following, we give a brief description of the CSP, and describe the weak-commitment search algorithm. Then, we show several empirical results indicating the efficiency of this algorithm. Furthermore, we examine the algorithm complexity and show a probabilistic model of the min-conflict backtracking and the weak-commitment search.

Constraint Satisfaction Problem

A constraint satisfaction problem can be described as follows. There exist n variables $x_1, x_2, ..., x_n$, each of which takes its value from a finite, discrete domain $D_1, D_2, ..., D_n$, respectively. There also exists a set of constraints. In this paper, we assume that a constraint is represented as a *nogood*, i.e., a combination of variable values that is prohibited[1]. We represent the fact that variable x_i's value is d_i as a tuple (x_i, d_i). A constraint $\{(x_i, d_i), (x_j, d_j)\}$ means that the combination of $x_i = d_i$ and $x_j = d_j$ is prohibited. A solution of a CSP is the value assignment of all variables that satisfies all constraints, i.e., the assignment that is not a superset of any nogood.

Weak-commitment Search Algorithm

The weak-commitment search algorithm is illustrated in Figure 1. Initially, *vars-left* is set to $\{(x_1, d_1), (x_2, d_2), \ldots, (x_n, d_n)\}$, where d_i is the tentative initial value of x_i. Also, *partial-solution* is assigned to an empty set. This algorithm moves variables from *vars-left* to *partial-solution* one by one.

The essential difference between this algorithm and the min-conflict backtracking is the shaded part in Figure 1. In min-conflict backtracking, backtracking is performed at this part and the most-recently added variable is removed from the partial solution. In weak-commitment search, the whole partial solution is abandoned, i.e., all elements of *partial-solution* are moved to *vars-left*. Then, the search process is restarted using the current value assignment as new tentative initial values. It must be noted that not all variable values in the partial solution will be revised again. Since the algorithm revises only the constraint violating variables, the variables that already satisfy all constraints will not be revised again.

This algorithm records the abandoned partial solutions in *nogoods* as new constraints, and avoids creating the same partial solution that has been created and abandoned before. Therefore, the completeness of the algorithm (always finds a solution if one exists, and terminates if no solution exists) is guaranteed.

In Ginsberg & Harvey (1990), a backtrack-based algorithm called the *iterative broadening* algorithm is presented. This algorithm avoids strong commitments and revises bad decisions without exhaustive search. The weak-commitment algorithm is similar to the iterative broadening algorithm where the search width is set to 1. However, the iterative broadening algorithm iteratively widens the search width if the trial employing the initial width fails. On the other hand, the weak-commitment search algorithm restarts the search process without changing the search width.

We show an example of algorithm execution using the well-known n-queens problem, placing n queens on an $n \times n$ chess board so that these queens will not threaten each other. In this example, we use the 4-queens problem. This problem can be formalized as a CSP where each variable represents the position of

[1] In general, a constraint is represented as an *allowed* combination of variable values. In this paper, we use the opposite representation since this representation is convenient for treating an abandoned partial solution as a new constraint. This choice of representation is inessential and does not affect the evaluation results in this paper.

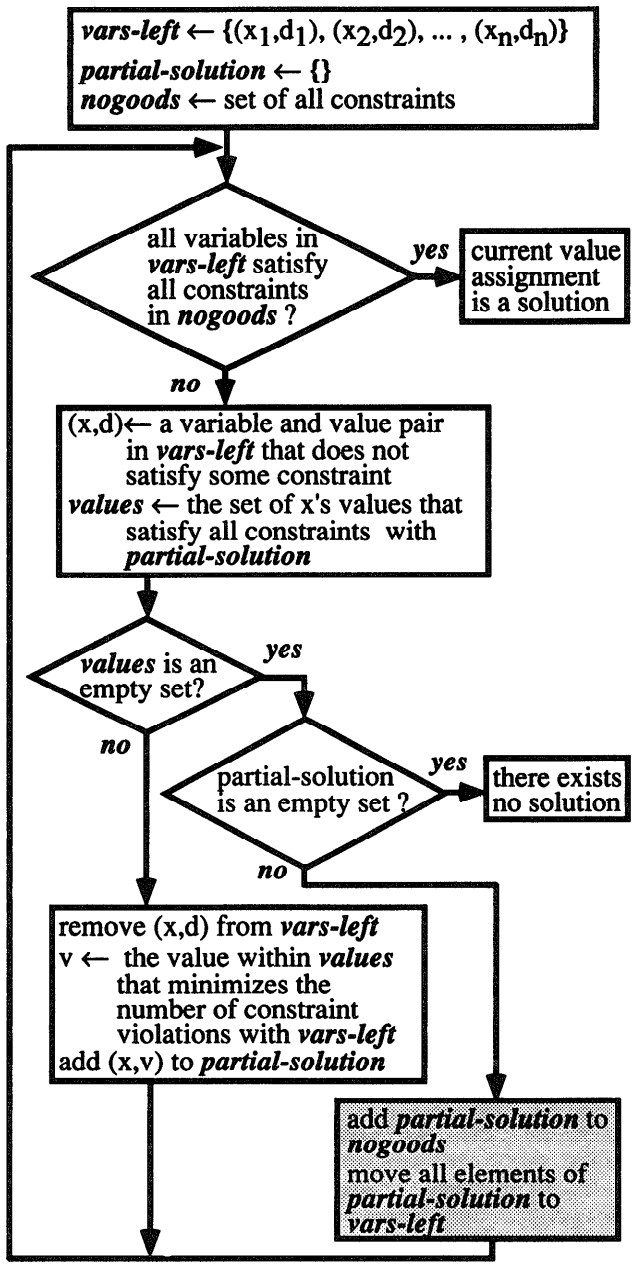

Figure 1: Weak-commitment search algorithm

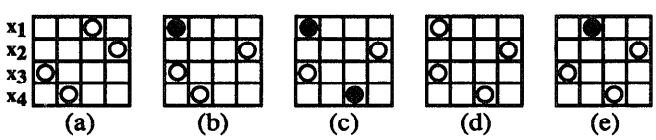

Figure 2: Example of algorithm execution

a queen in each row, and the domain of a variable is {1,2,3,4}. The initial state is illustrated in Figure 2(a). The algorithm first revises the position of the first queen (Figure 2(b)), then revises the position of the fourth queen (Figure 2(c)). A queen whose position is revised is added to *partial-solution*. We represent a queen in *partial-solution* as a filled circle. In Figure 2(c), there exists no consistent position with *partial-solution* for the third queen. Therefore, the whole *partial-solution* is abandoned (Figure 2(d)). The algorithm revises the position of the first queen again (Figure 2(e)). Consequently, all constraints are satisfied.

Evaluations

In this section, we compare the weak-commitment search, the min-conflict backtracking, and an iterative improvement algorithm by experiments on typical examples of CSPs (n-queens, graph-coloring, and 3-SAT problem).

We use the breakout algorithm (Morris 1993) as the representative for iterative improvement algorithms. This algorithm has a notable feature in that it does not stop at local minima, and has been shown to be more efficient than other iterative improvement algorithms (Morris 1993). However, this algorithm is not complete, i.e., the algorithm can fall into an infinite processing loop.

We measure the number of required steps and the number of consistency checks. Each change of one variable value, each backtracking, and each restarting is counted as one step. Also, one consistency check represents the check of one combination of variable values among which a constraint exists[2]. The number of consistency checks is widely used as a machine-independent measure for constraint satisfaction algorithms. For all three algorithms, in order to reduce unnecessary consistency checks, the result of consistency checks at the previous step is recorded and only the difference is calculated in each step.

In order to terminate the experiments in a reasonable amount of time, the maximum number of steps is limited to 5000, and we interrupt any trial that exceeds this limit. For an interrupted trial, we count the number of required steps as 5000, and use the number of consistency checks performed until the interruption for the evaluation.

n-queens

The first example problem is the n-queens problem described in the previous section. We show the ratio of successful trials (trials finished within the limit), the number of required steps, and the number of consistency checks for n=10, 50, 100 in Table 1. We run 100 trials with different initial values and show the

[2]The number of checks for newly added constraints (abandoned partial solutions) is also included.

n	weak-commitment			min-conflict BT			breakout		
	ratio	steps	checks	ratio	steps	checks	ratio	steps	checks
10	100%	29.7	2292.8	100%	240.7	15482.2	100%	41.7	7065.0
50	100%	23.9	48593.5	97%	264.5	300175.0	100%	37.9	180393.5
100	100%	27.1	236821.7	99%	76.4	912465.1	100%	38.9	777051.1

Table 1: Comparison on n-queens

n	ratio of trials with BT	weak-commitment	min-conflict BT
10	80%	35.9	299.7
50	17%	58.2	1473.4
100	2%	96.5	2563.5

Table 2: Required steps for trials with backtracking/restarting

average[3]. These initial values are generated by the greedy method described in Minton *et al.* (1992).

As shown in Table 1, for all cases, the weak-commitment search is more efficient than the min-conflict backtracking and the breakout algorithm. We can see that the breakout algorithm does a lot more consistency checks for each step compared with the weak-commitment search. This fact can be explained as follows. When choosing a variable to change its value, the weak-commitment search (and the min-conflict backtracking) can choose any of the constraint violating variables. On the other hand, the breakout algorithm must choose a variable so that the number of constraint violations can be reduced by changing its value. Therefore, in the worst case (when the current assignment is a local minimum), the breakout algorithm has to check all the values for all constraint violating variables[4].

For the trials without backtracking/restarting, the behaviors of the weak-commitment search and the min-conflict backtracking are exactly the same. We show the ratio of trials with backtracking/restarting, and the average number of steps for these trials in Table 2. We can see that the numbers of required steps for the min-conflict backtracking in trials with backtracking are very large and dominate the average of all trials.

[3]In Minton *et al.* (1992), it is reported that the min-conflict backtracking can solve the 100-queens problem in around 25 steps. In our experiment, there exists one trial that exceeds 5000 steps and the result of this trial becomes the dominant factor in the average. The average except this trial is almost identical to the result reported in Minton *et al.* (1992). For $n \geq 1000$, the result for the min-conflict backtracking and weak-commitment search are exactly the same, and almost identical to the result reported in Minton *et al.* (1992).

[4]If the current assignment is not a local minimum, the breakout algorithm does not have to check all variables.

Graph-coloring

The graph-coloring problem involves painting nodes in a graph by k different colors so that any two nodes connected by an arc do not have the same color. We randomly generate a problem with n nodes and m arcs by the method described in Minton *et al.* (1992), so that the graph is connected and the problem has a solution. We evaluate the problem $n = 120, 180, 240$, where $m = n \times 2$ and k=3. This parameter setting corresponds to the "sparse" problems for which Minton *et al.* (1992) report poor performance of the min-conflict heuristic. We generate 10 different problems, and for each problem, 10 trials with different initial values are performed (100 trials in all). As in the n-queens problem, the initial values are set by the greedy method.

We introduce two kinds of heuristics (*forward checking* and *first-fail principle* (Haralick & Elliot 1980)) into the min-conflict backtracking and the weak-commitment search, i.e., for each variable, we keep the list of values consistent with the partial solution, and when selecting a variable to be added to the partial solution, we select the variable that has the least number of consistent values. Also, the variable that has only one consistent value is included into the partial solution immediately. Furthermore, before including a variable into the partial solution, we check whether each of remaining variables (variables in *vars-left*) has at least one consistent value with the partial solution, and avoid selecting a value that causes immediate failure. Table 3 shows evaluation results for the three algorithms.

Although Minton *et al.* (1992) report poor performance for the min-conflict backtracking for these problems, by introducing the two heuristics (*forward checking* and *first-fail principle*), the performance of the min-conflict backtracking becomes relatively good in our evaluation. However, there exist several trials in which a mistake in the value selection becomes fatal, and the min-conflict backtracking fails to find a solution within the limit. Therefore, the weak-commitment search is more efficient than the min-conflict backtracking.

For these problems, the *forward checking* and *first-fail principle* are very efficient and the weak-commitment search is about 10 times more efficient than the breakout algorithm. We can see that the capability of accommodating such powerful heuristics is a great advantage of the weak-commitment search over iterative improvement algorithms.

n	weak-commitment			min-conflict BT			breakout		
	ratio	steps	checks	ratio	steps	checks	ratio	steps	checks
120	100%	28.9	2118.8	99%	78.1	2931.2	100%	198.4	14620.8
180	100%	41.3	3178.9	99%	98.5	5548.5	100%	352.3	32139.3
240	100%	71.9	5988.6	95%	443.6	42164.5	100%	601.2	66892.5

Table 3: Comparison on graph-coloring

n	weak-commitment			min-conflict BT			breakout		
	ratio	steps	checks	ratio	steps	checks	ratio	steps	checks
300	100%	187.7	24357.0	55%	2717.7	1075986.1	100%	828.0	133911.5
500	100%	359.4	47376.0	15%	4428.8	2368672.5	93%	1596.2	352095.8
700	100%	633.2	83345.1	0%	—	—	77%	2740.2	746752.5
900	100%	980.3	132731.7	0%	—	—	70%	3006.8	1032267.3
1100	100%	1246.8	168845.9	0%	—	—	69%	3384.5	1454297.9

Table 4: Comparison on 3-SAT problem

3-SAT Problem

For the third example problem, we use the 3-SAT problem, which is commonly used as a benchmark for iterative improvement algorithms. This problem is to assign truth values for n boolean variables that satisfy constraints represented as clauses. Each clause consists of three variables. The number of clauses divided by the number of variables is called the *clause density*, and the value 4.3 has been identified as the critical value that produces particularly difficult problems (Mitchell, Selman, & Levesque 1992).

Table 4 shows results of changing the number of variables n by setting the clause density to 4.3. We use the same method described in Morris (1993) to generate a random problem that has a solution. We generate 10 different problems, and for each problem, 10 trials with different initial values[5] are performed (100 trials in all). As in the case of the graph-coloring problems, we introduce the two heuristics into the min-conflict backtracking and the weak-commitment search.

As shown in Table 4, the min-conflict backtracking is very inefficient for this problem. On the other hand, the weak-commitment search is around 10 times more efficient than the breakout algorithm for larger n. For this problem, the effect of the heuristics is not powerful enough to completely avoid bad decisions. Such less powerful heuristics are of little use to the min-conflict backtracking. On the other hand, they are useful for the weak-commitment search algorithm since the variable values are iteratively revised by restarting.

Discussions

Algorithm Complexity

The worst-case time complexity of the weak-commitment search becomes exponential in the number of variables n. This result seems inevitable since constraint satisfaction is NP-complete in general. The space complexity of the weak-commitment search is determined by the number of newly added nogoods (constraints) to *nogoods*, i.e., the number of restartings. In the worst case, this is also exponential in n. On the other hand, the space complexity of the backtracking algorithm is linear to n. This result seems inevitable since the weak-commitment search changes the search order flexibly while guaranteeing the completeness of the algorithm. This is also the case for the fill algorithm described in Morris (1993), whose worst-case space complexity becomes exponential in n.

However, the number of restartings will never exceed the number of required steps. Therefore, we can assume that the space complexity would never become a problem in practice as long as the problem can be solved within a reasonable amount of time. Also, the nogood that is a superset of other nogoods is redundant and can be removed from *nogoods*.

Furthermore, we can restrict the number of recorded nogoods. In that case, the theoretical completeness can not be guaranteed. However, in practice, the weak-commitment search algorithm can still find a solution for all example problems when the number of recorded nogoods is restricted so that only 10 of the most recently found nogoods are recorded.

Probabilistic Model

In order to show theoretical evidence that the weak-commitment search is more efficient than the min-conflict backtracking, we use a simple probabilistic model as follows. Let us assume that the probability for finding a solution without any backtracking in the min-conflict backtracking is given by the constant value p, regardless of the initial values. Also, let us assume that the expected number of steps for trials without backtracking is given by n_s ($n_s \leq n$), the ex-

[5]These initial values are set by the greedy method as in the case of the other problems.

pected number of steps for trials with backtracking is given by B, and the expected number of steps until the occurrence of the first backtracking is given by n_b ($n_b \leq n$). Then, the expected number of steps for the min-conflict backtracking can be represented as $n_s p + Bq$, where $q = 1 - p$.

On the other hand, in the weak-commitment search, a solution can be found without any restarting with the probability p, and the expected number of steps in this case is given by n_s. Also, the probability that a solution can be found after one restarting is given by pq, and the expected number of steps is given by $n_s + n_b$. In the same way, the probability that a solution can be found after k restartings is given by pq^k, and the expected number of steps is given by $n_s + kn_b$. This probability distribution of the number of restartings is identical to the well-known geometric distribution, and the expected number of restartings is given by q/p. Therefore, the expected number of steps can be given by $n_s + n_b q/p$.

The condition that the weak-commitment search is more efficient than the min-conflict backtracking is $n_s p + Bq > n_s + n_b q/p$. By transforming this formula, we obtain $p > n_b/(B - n_s)$. Since we can assume that $B \gg n_s$, and $n_b \leq n$, we obtain the sufficient condition $p > n/B$, i.e., if the probability of finding a solution without backtracking is larger than the ratio of the number of variables and the number of steps for the trials with backtracking, the weak-commitment search is more efficient than the min-conflict backtracking.

The experimental results in the previous section show that the min-conflict backtracking can find a solution efficiently without backtracking in many trials, but the number of required steps for trials with backtracking becomes very large. Therefore, we can assume that the condition $p > n/B$ is satisfied in many cases. For example, from the experimental results in Table 2, we can assume that B for the 50-queens problem is around 1473.4, and p is around 0.83 (where q is 0.17). Then, n/B is around 0.034. This value is much smaller than the expected value of p (0.83).

In reality, the probability of finding a solution without backtracking is affected by the initial values. In the weak-commitment search, when restarting, the current value assignment is used as the new tentative initial values. Therefore, by repeating the restartings, we can expect the value assignment to become close to the final solution; thus, the probability of finding a solution without restartings increases. In such a case, even if the average probability of finding a solution without backtracking is very small and the condition $p > n/B$ is not satisfied, the weak-commitment search can be more efficient than the min-conflict backtracking. For example, although the min-conflict backtracking never finds a solution without backtracking in the 3-SAT problem (Table 4), the weak-commitment search can find a solution. This is because the value assignment is iteratively improved in the weak-commitment search.

Conclusions

We have presented the *weak-commitment* search algorithm for solving CSPs. This algorithm can revise bad decisions without exhaustive search, and the completeness of the algorithm is guaranteed. By experimental results, we have shown that this algorithm is 3 to 10 times more efficient than the breakout algorithm and the min-conflict backtracking.

Our future work includes showing the effectiveness of this algorithm in practical application problems, developing a theoretical model to compare the performance of the weak-commitment search and iterative improvement algorithms, and applying the weak-commitment search algorithm to *distributed constraint satisfaction problems*, in which variables and constraints are distributed among multiple problem solving agents (Yokoo *et al.* 1992).

Acknowledgments

The author wishes to thank Tsukasa Kawaoka, Ryohei Nakano, and Nobuyasu Osato for their support in this work at NTT Laboratories; and Toru Ishida, Kazuhiro Kuwabara, Shigeo Matsubara, and Jun-ichi Akahani for their helpful comments.

References

Ginsberg, M. L., and Harvey, W. D. 1990. Iterative broadening. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 216–220.

Haralick, R., and Elliot, G. L. 1980. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence* 14:263–313.

Mackworth, A. K. 1992. Constraint satisfaction. In S.C.Shapiro., ed., *Encyclopedia of Artificial Intelligence*. Wiley-Interscience Publication. 285–293.

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

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and easy distributions of SAT problem. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 459–465.

Morris, P. 1993. The breakout method for escaping from local minima. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 40–45.

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 440–446.

Yokoo, M.; Durfee, E. H.; Ishida, T.; and Kuwabara, K. 1992. Distributed constraint satisfaction for formalizing distributed problem solving. In *Proceedings of the Twelfth IEEE International Conference on Distributed Computing Systems*, 614–621.