Anytime Algorithms for Constraint Satisfaction and SAT problems*

Richard J. Wallace and Eugene C. Freuder
Department of Computer Science
University of New Hampshire
Kingsbury Hall Rm 208, College Road
Durham, NH 03824 USA
rjw,ecf@cs.unh.edu

Abstract

The constraint satisfaction problem (CSP) is a potential area of application for anytime methods. In this work, we derive anytime curves using a partial constraint satisfaction framework that encompasses problems with complete solutions and those that allow only partial solutions of varying quality. In either case, the curves should converge on optimal solutions with respect to some measure of cost (here, the number of violated constraints). Binary CSPs and k-satisfiability problems were tested, using heuristic repair and branch and bound methods. Curves for heuristic methods either start at a lower level than curves for branch and bound (min-conflicts with binary CSPs) or have a steeper initial descent (GSAT with k-SAT problems). Techniques for randomization such as random walks or restarting with a new random solution appear to be necessary with heuristic procedures for complete convergence to an optimal solution. Branch and bound algorithms are usefully employed in tandem with heuristic methods, especially to verify optimality and, therefore, the quality of solution returned by the latter.

1 Introduction

For problems that are inherently difficult to solve, anytime methods provide a means of obtaining imperfect but useful solutions within a reasonable time frame. The constraint satisfaction problem (CSP) is a type of hard problem that has found application in numerous areas of engineering and design. This problem, therefore, constitutes an important potential area of application for anytime methods.

Constraint satisfaction problems (CSPs) involve finding an assignment of values to variables that satisfy a set of constraints between these variables. In order to apply anytime methods to these problems, the latter must be placed within a more general framework, in which partial solutions are considered as well as complete solutions. A form of partial constraint satisfaction (PCSP) for which anytime methods are well-suited is the maximal constraint satisfaction problem (MAX-CSP), in which the goal is to find assignments of values to variables that satisfy the maximum number of constraints. In this form of the problem, an anytime procedure should find successively better solutions, satisfying a greater number of constraints, as time progresses.

By considering CSPs as PCSPs, we can also treat problems that are overconstrained, so that no complete solution is possible, within the same framework as fully solvable CSPs. For

overconstrained problems, anytime methods should converge on incomplete, optimal (i.e., maximal) solutions.

An important type of CSP is the satisfiability problem (SAT), in which the goal is to find a truth assignment that satisfies a CNF formula in propositional logic. A more general form of SAT that is suited to anytime methods is the maximum satisfiability problem (MAX-SAT), in which the object is to find a truth assignment that satisfies a maximum number of clauses in the formula. Here, anytime methods should find assignments that satisfy more clauses (equivalent to satisfying more constraints) as time progresses.

In this work we study the anytime properties of two kinds of procedure that can return incomplete or "partial" solutions of increasing quality over time, whether or not the problem has a complete solution. The first is a type of branch and bound algorithm designed for CSPs. Search algorithms designed for SAT (e.g., the Davis-Putnam procedure) can be similarly extended to handle MAX-SAT problems. The second is a type of heuristic procedure that begins with a complete assignment of values that violates an unspecified number of constraints, and then tries to repair the initial solution to reduce the number of errors. The latter procedure does not find a succession of solutions that converge toward the optimum in a strict sense. But the requisite anytime properties can be obtained by saving the best solution found so far, if, as time progresses, solutions are discovered that are better than any earlier solution. The present results show that this is the case, and that characteristic (and distinct) curves are found for both methods of search.

2 Algorithms

A constraint satisfaction problem (CSP) involves assigning values to variables that satisfy a set of constraints among subsets of these variables. The set of values that can be assigned to one variable is called the domain of that variable. In the present work all constraints are binary, i.e., they are based on the Cartesian product of the domains of two variables. A binary CSP is associated with a constraint graph, where nodes represent variables and arcs represent constraints. If two values assigned to variables that share a constraint are not among the acceptable value-pairs of that constraint, this is an inconsistency, or conflict, or constraint violation. The relative number of inconsistent value-pairs is called the tightness of the constraint.

Algorithms for CSPs that employ exhaustive search try to extend partial solutions to complete solutions, while minimizing unproductive search by backtracking and establishing forms of local consistency, i.e., consistency within subsets of variables. Branch and bound versions of these algorithms use the number of inconsistencies in a partial solution as a cost function to limit search; if this number equals or exceeds the number of inconsistencies in the best solution found so far, then the partial solution will not lead to a better solution and search can back up at this point.

* This material is based on work supported by the National Science Foundation under Grant No. IRI-9207633 and IRI-9504316.

©1996 R. J. Wallace and E. C. Freuder
In the present work we used a branch and bound version of forward checking, a well-known CSP algorithm. Different versions of this algorithm were used for different problem sets; the choice was based on extensive study of these algorithms on problems of different types. The goal was to test an exhaustive search algorithm that does relatively well on the given type of problem. For sparse problems we used forward checking with variables ordered by degree of node in their constraint graph (highest degree first); if the constraints were tight, this was augmented by preprocessing strategies in combination with search that take better advantage of inconsistencies and allow us to order values within a domain so that those associated with fewer inconsistencies are chosen first. For denser problems we used forward checking with dynamic variable ordering based on current domain size (smallest domain first). For MAX-SAT problems we used a branch and bound version of the Davis-Putnam algorithm that includes the in-most-shortest clause heuristic for choosing the next variable to assign. (These algorithms are described in detail in [FW92], [Sas81], [Wal95], [WF95].)

Heuristic repair procedures for CSPs begin with a complete assignment and try to improve it by choosing alternative assignments that reduce the number of constraint violations. In the minconflicts procedure, the first assignment is made by choosing values that minimize the number of constraint violations with values already chosen ([MJL92]). Then random methods are used to choose a variable whose assignment conflicts with another assignment and, from the domain of that variable, a value that has the minimal number of conflicts. GSAT, a repair procedure for SAT problems, begins with a random truth assignment and then alters ("flips") the assignment of a variable that leads to the greatest increase in satisfiable clauses. After a certain number of flips, a new random assignment is chosen (called a new "try") ([SLM92]). A variation of GSAT, here called walk-SAT, flips the value of some variable chosen at random with a certain probability. Two features of this procedure were also tested with minconflicts: retries after a certain number of new assignments and the random walk strategy. In these studies the number of assignments before a retry was five times the number of variables, while the probability of a random choice in the walk procedure was always 0.35. These values are based on values used in published reports.

### 3 Experimental Methods

Random constraint satisfaction problems were generated using the "probability of inclusion" model of generation (cf. [FW92]). The number of variables was fixed, as well as the maximum domain size. Possible domain elements, constraints and constraint value pairs were each chosen with a specified probability.

The initial sets of problems had 30 variables and a domain maximum of 10. Probability values were fixed to give an expected domain size of 5, an expected density of 0.05 (in terms of the number of edges added to a spanning tree), and an expected tightness of 0.50, 0.70, or 0.80. Other problems had the same number of variables, with many more constraints (expected density = 0.33); these problems had an expected domain size of 8 (max = 16), and an expected tightness of 0.70. Problems with density = 0.5 and tightness = 0.50 are near the transition region where the likelihood of a solution drops dramatically; about 25 percent of those in the sample had solutions. None of the other problems had solutions. All of these problems could be run to completion with the branch and bound methods used, so that quality of solution could be evaluated by comparisons with a known optimal distance as well as with solutions obtained by other methods.

![Figure 1: Averaged anytime curves for branch and bound and heuristic repair algorithms. 30-variable problems, exp. density = 0.05, exp. tightness = 0.50.](image)

Other sets of problems had 100 variables, an expected density of 0.02, expected domain size of 5 (max = 10), and an expected tightness of 0.70. These problems also were near the transition region as described above. Problems of this size are not easily solved by branch and bound methods, although in some cases it has been possible to obtain verified optimal solutions.

3-SAT problems were produced with a program obtained from Bell Laboratories. This generates random SAT problems with a specified number of variables and fixed-length clauses. Problems had 50 variables and either 225 or 300 clauses. In the first case the problems are near the transition region described above; about 25 percent of the sample problems had solutions. None of the 300-clause problems had solutions. Problems with these features can be solved to completion with branch and bound methods, so that the quality of solution returned by the heuristic methods could be evaluated in relation to the known optimum, as well as in relation to the complete method.

For all sets of problems, the sample size was 25. In some cases ten successive runs were made through the entire sample to determine the stability of the mean for number of inconsistencies after k sec. For each point tested, the standard deviation of the means was about 5% of their mean value.

### 4 Results: MAX-CSPs

Anytime curves for the mean number of constraint violations in the best solutions found after varying periods of time are shown in Figure 1 for branch and bound (forward checking) and heuristic repair (min-conflicts). These data are based on relatively small problems (n = 30) with sparse constraint graphs (expected density = 0.05) and moderately tight constraints (expected tightness = 0.50). Both curves are decelerating, but the curve for min-conflicts shows a transition from the decelerating phase to a stable plateau, while the curve for forward checking declines, albeit at a diminishing rate, throughout the period shown. As a result, although the curve for min-conflicts is initially much lower than that for the branch and bound algorithm, the latter eventually
catches up, so that after a few seconds, the average solution found by branch and bound is somewhat superior to that found by the heuristic method. In this case optimal solutions were found for all 25 problems by forward checking by the end of the 10-second period, while min-conflicts had found only 6 optimal solutions. (And there was no further improvement in the remaining 90 sec of testing.)

For problems with tighter constraints the same trends were found, but the crossover point occurred after longer durations: at about 1 min for problems with expected tightness = 0.70 and more than 100 sec for problems with expected tightness = 0.80. In the first case, after 1 min forward checking had found 19 optimal solutions, min-conflicts, 7; in the latter case, at 100 sec forward checking had found only 5 optimal solutions, and min-conflicts had found 10.

For expected tightness = 0.80, forward checking is effectively enhanced with preprocessing techniques. In this case the curves crossed at about 50 sec, and 11 optimal solutions had been found by the branch and bound algorithm by this time, while min-conflicts had found 10. (However, these 10 were all found within 10 sec.)

Using the same preordering heuristic for min-conflicts as was used for forward checking (maximum degree) resulted in a small improvement in the initial distances, but the difference was gone after 0.05 sec. Similar results were found for other sets of problems with degree and domain size heuristics, so it appears that preordering is not an effective strategy with these procedures.

In contrast, both the random walk and the retry procedures enhanced the effectiveness of min-conflicts. For these values of walk and retry, the former strategy was more effective than the latter for problems with looser constraints (expected tightness = 0.50); in this case, the anytime curve descended until all optimal solutions had been found at about 10 sec. However, this variant was inferior to the basic min-conflicts procedure for the two sets of problems with tighter constraints. In these latter cases, the retry strategy was superior to the basic procedure, most dramatically for the problems with the tightest constraints. In this case 22 optimal solutions were found by 50 sec (see Figure 2).

For 30-variable problems with higher density (expected value = 0.33) a similar pattern of results was found for branch and bound (in this case, forward checking with dynamic search rearrangement) and min-conflicts. Although the average optimal distance was 0.92, min-conflicts remained at 3.92 for the duration of testing (100 sec). Without further improvement in the curve for min-conflicts (which seems likely based on earlier results), the curves would cross between 100 and 500 sec. For these problems the retry procedure was superior to the other variants, and the curve reached a value of 1.28 for the mean distance by 100 sec.

Figure 3: Averaged anytime curves for branch and bound and heuristic repair algorithms. 100-variable problems, exp. density = 0.02, exp. tightness = 0.60. Upper curve is forward checking.

For the 100-variable problems, branch and bound was unable to find good solutions even after 1000 sec. For this algorithm, the mean distance of the best solutions found in this time was 23.0. In contrast, the curve for the basic min-conflicts reached a value of 4.08 after 5 sec, although no further improvement was found in the next 95 sec. For these problems the walk variant was the best; the curve for this procedure reached a value of 0.96 by 500 sec. (Optimality is being tested using the values found by this procedure as initial upper bounds; to date all completed runs have verified that the solution found by this heuristic repair procedure is optimal.)

5 Results: MAX-SAT

For 50-variable problems with 225 clauses, Davis-Putnam, GSAT and walk-SAT all found optimal solutions to all problems within 100 sec. The anytime curves were similar to those found for MAX-CSP algorithms: a steeper initial descent for the heuristic procedures and eventual convergence. (Since GSAT does not try to minimize conflicts in its initial assignment, the curves in this case all begin at about the same value.) For problems with 300 clauses the differences were similar, but the time required for convergence now exceeded 100 sec. In this time the branch and bound algorithm found and verified optimal solutions for 12 problems. In each of these cases, walk-SAT and GSAT also found optimal solutions.

6 Conclusions

Anytime curves for constraint satisfaction problems are always decelerating, with a steep initial slope. (Preliminary results with nonlinear regression analysis suggest that, based on a simple negative exponential model, the averaged curves are best fitted with two equations, one with a time constant $\gg 1$ and one with a time constant $\ll 1$.) Heuristic procedures
differ from branch and bound in having a lower initial value (min-conflicts) and/or a steeper initial decline (e.g., GSAT, cf. [SK93]). These differences are greatly amplified as problem size increases. In fact, at 100 variables the decelerating character of the curve for branch and bound is no longer obvious over short intervals (Figure 3).

For the basic min-conflicts procedure the anytime curve quickly descends to a stable plateau, which presumably means that the procedure is stuck at a local optimum. Strategies of random selection allow the procedure to escape such local optima, so that the anytime curve converges on the global optimum. For branch and bound, the initial rate of decline can be enhanced in some cases by preprocessing techniques (Figure 2).

These studies suggest that the two approaches for solving CSPs can profitably be used in tandem. There are two situations where this may be useful. For small problems, in which hill-climbing finds the better solutions at first, but where branch and bound algorithms can sometimes overtake the former, running the algorithms in parallel or in a time-sharing framework may yield a better quality solution after time $t$ than either algorithm alone. For larger problems, this procedure is not likely to improve on hill-climbing alone. However, if certification of optimality is important, then a composition of both strategies, as described in [Zil95], may be useful. In this case, hill-climbing can be run first, giving a good solution which is used to set the initial upper bound for the branch and bound procedure. Using techniques like those described in [GZ95], it should be possible to determine the best durations to run each procedure for a given set of problems.

References


Figure 4: Averaged anytime curves for Davis-Putnam (branch and bound) and GSAT procedures. 50-variable, 225-clause problems.