Conjunctive Width Heuristics for Maximal Constraint Satisfaction

Richard J. Wallace and Eugene C. Freuder*

Department of Computer Science

University of New Hampshire, Durham, NH 03824 USA

rjw@cs.unh.edu; ecf@cs.unh.edu

Abstract

A constraint satisfaction problem may not admit a complete solution; in this case a good partial solution may be acceptable. This paper presents new techniques for organizing search with branch and bound algorithms so that maximal partial solutions (those having the maximum possible number of satisfied constraints) can be obtained in reasonable time for moderately sized problems. The key feature is a type of variable-ordering heuristic that combines width at a node of the constraint graph (number of constraints shared with variables already chosen) with factors such as small domain size that lead to inconsistencies in values of adjacent variables. Ordering based on these heuristics leads to a rapid rise in branch and bound's cost function together with local estimates of future cost, which greatly enhances lower bound calculations. Both retrospective and prospective algorithms based on these heuristics are dramatically superior to earlier branch and bound algorithms developed for this domain.

1 Introduction

Constraint satisfaction problems (CSPs) involve finding values for problem variables subject to restrictions on which combinations of values are allowed. They are widely used in AI, in areas ranging from planning to machine vision. Many CSP applications settle for partial solutions, where some constraints remain unsatisfied, either because the problems are overconstrained or because complete solutions require too much time to compute. In fact, such applications generally settle for suboptimal partial solutions; obtaining a solution optimally close to a complete solution can be extremely difficult even for small problems (Freuder & Wallace 1992). In this paper we describe techniques that permit solving many moderately sized optimization problems within practical time bounds. (Smaller problems can be solved quickly enough for real-time applications.)

Maximal constraint satisfaction problems require solutions that optimize the number of satisfied constraints. Branch and bound methods (cf. Reingold et al. 1977) can be combined with constraint satisfaction techniques to find maximal solutions (Freuder & Wallace 1992) and, unlike hill climbing techniques, branch and bound can guarantee an optimal solution. This paper presents new search order heuristics for branch and bound maximal constraint satisfaction search. These heuristics permit formulation of algorithms that are in many cases markedly superior to previously studied branch and bound maximal constraint satisfaction algorithms.

The design and application of these heuristics embody three key features (using concepts discussed at greater length in subequent sections):

**Conjunctive ordering heuristics based on width*: Width at a node in an ordered constraint graph has been shown to be an effective heuristic for CSPs (Dechter & Meiri 1989; there called "cardinality"). Here we show that combining this with other heuristic factors can produce conjunctive heuristics whose power is "greater than the sum of their parts". This is because, in addition to providing the successive filtering that would be expected to improve performance, certain heuristics function synergistically with width to yield a marked reduction in the search space through effective tightening of bounds.

**Use of information gained in preprocessing*: The effectiveness of these algorithms also depends strongly on measures of arc (in)consistency obtained for each value of each variable during preprocessing. This information is gained cheaply, in one pass through the problem before search begins. It can then used for ordering both values and variables as well as in the calculation of bounds to determine whether a given value is selected during search. While the latter procedure can only be used with retrospective algorithms, e.g., backmarking, ordering based on these measures can also be used with prospective algorithms such as forward checking.

**Effective lower bound calculation*: Earlier analyses of techniques for calculating lower bounds on the cost of a solution (Freuder & Wallace 1992; Shapiro & Haralick 1981) did not explore the means by which a given component of this calculation could be maximized. In particular, they did not consider the possibility of increasing the cost at the current node of the search tree as rapidly as possibly while *at the same time* maximizing

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

the components of the lower bound that are based on future (uninstantiated) variables. Conjunctive width heuristics promote this dual effect, thus providing a kind of 'one-two punch' that raises the lower bound quickly enough to enhance pruning dramatically. From another viewpoint, this is an extension of the Fail First principle (Haralick & Elliott 1980) to the more subtle problem of lower bound calculation. Somewhat surprisingly, for some classes of problems this allows a retrospective algorithm based on backmarking and local lower bound calculations (RPO, [Freuder & Wallace 1992]) to outperform a prospective algorithm, forward checking, that uses extended lower bound calculations based on all future variables.

In the remaining sections we elucidate these properties and investigate them experimentally. In particular, we present:

(i) a set of experiments with sparse random problems that demonstrates the marked superiority of the new algorithms over the best maximal constraint satisfaction algorithms tested previously and elucidates the basis of this superior performance.

(ii) a second set of experiments that examines the range of problem parameters for which the new algorithms are superior. The parameters of greatest interest are the relative number of constraints between variables and the average *tightness* of the constraints (a tight constraint has few acceptable value-pairs).

Section 2 describes the basic features of all algorithms tested in this study. Section 3 describes results of experiments with an initial set of sparse problems. Sections 4 and 5 analyze the factors that make the new algorithms effective. Section 6 examines the range of problem parameters over which the new algorithms are superior to others. Section 7 gives our conclusions.

2 Description of the Algorithms

The algorithms discussed in this paper are all based on depth-first search with backtracking. These algorithms try to find a value for each variable in the problem drawn from its *domain*, or set of allowable values, so that the number of constraints among variables that cannot be satisfied is minimized. (In this work only binary constraints are considered.) Since these are branch and bound algorithms, they use a cost function related to constraint failure; here, this is simply the number of violated constraints in the solution, called the *distance* from a complete solution. This number is compared with the distance of the best solution found so far to determine whether the current value can be included in the present partial solution. The distance of this best solution is, therefore, an *upper bound* on the allowable cost, while the distance of the current partial solution is an elementary form of *lower bound* (i.e., the cost of a solution that includes the values chosen so far cannot be less than the current distance).

Implementing this strategy in its simplest form results in a basic branch and bound algorithm, P-BB. More sophisticated strategies are possible that are analogous to those used to solve CSPs. Here, we consider, (i) a backmarking analogue (P-BMK) that saves the increment in distance previously derived for a value, analogous to the "mark" stored by ordinary backmark; this increment can be added to the current distance to limit unnecessary repetition of constraint checking, (ii) a forward checking analogue (P-EFC) in which constraint failures induced by values already selected are counted for each value of the future variables and can be added to the current distance to determinine whether the bound is exceeded; in addition, the sum of the smallest counts for each domain of the uninstantiated variables (other than the one being considered for instantiation) is added to the current distance to form a lower bound based on the entire problem. In this paper we are most concerned with versions of P-EFC which incorporate some form of dynamic search rearrangement. (These algorithms are described more fully in [Freuder & Wallace 1992].)

P-BMK (as well as P-BB) can incorporate tallies of constraint violations based on arc consistency checking prior to search (ACCs, for "arc consistency counts"); specifically, the ACC for a given value is the number of domains that do not support it. These are used in lower bound calculations, much as the counts used in the forward checking analogues. In addition, the values of each domain can be ordered by increasing ACCs so that values with the most support are selected first. Freuder and Wallace (1992) call the resulting algorithm RPO, in reference to its retrospective, prospective and ordering components.

The new algorithms combine the procedures just described with variable orderings based on two or three factors. The first is always the width at a node of the *constraint graph*, that represents the binary relations between variables (represented as nodes). Width at a node is defined as the number of arcs between a node and its predecessors in an ordering of the nodes of a graph (Freuder 1982). The associated heuristic, maximum (node) width, is the selection of the variable with the greatest number of constraints in common with the variables already chosen, as the next to instantiate. In a conjunctive heuristic, ties in maximum width are broken according to a second heuristic, here, either minimum domain size, maximum degree of the node associated with the variable, or largest mean ACC for a domain. Ties in both factors can be broken with a third heuristic which is also one of those just mentioned. In subsequent sections, the order in which these heuristics are applied is indicated by slashes between them: thus, in the width/domain-size/degree heuristic, ties in maximum width are broken by choosing as the next variable one with the smallest domain size, and further ties are broken by choosing the node of largest degree. It is important to note that the first variable to be instantiated is always chosen according to the second or third heuristic, since at the beginning of search all widths are zero.

3 Initial Experiments

In our initial experiments we naturally wanted problems for which the new algorithms would be likely to excel. Earlier evidence suggested that local lower bound calculation would be especially effective with problems that had sparse constraint graphs. At the same time, it was expected that P-EFC would not do as well because, due to the sparseness of the problem, the minimum counts for most future domains would be zero. In addition, we wanted problems with marked heterogeneity in the size of their domains and constraints, to insure a fair amount of inconsistency between adjacent variables.

With these ends in mind, random problems were generated as follows. Number of variables (n) was fixed at either 10, 15 or 20. To obtain sparse problems, the number of constraints was set to be (n-1) + ceiling(n/2); therefore, the average degree of nodes in the constraint graph was three. Domain size and constraint satisfiability (number of acceptable value pairs) were determined for each variable and constraint by choosing a number between one and the maximum value allowed, and then choosing from the set of possible elements until the requisite number had been selected. The maximum domain size was set at 9, 12 or 15 for 10-, 15- and 20-variable problems, respectively. Note that domain sizes and constraint satisfiabilities will approximate a rectangular distribution, giving problems that are heterogeneous internally, but with have similar average characteristics. In addition, if two domains had only one element, there could be no effective constraint between them; to avoid this, domain sizes were chosen first, so that this condition could be disallowed when selecting constraints. Twenty-five problems were generated for each value of n. The mean distance for maximal solutions was always 1.2-1.6.

To measure performance we used constraint checks and nodes searched. When the two are correlated or when the former measure predominates, that measure is reported alone. Constraint checks done during preprocessing are always included in the total.

Figure 1 shows the mean performance on these problems for algorithms described in earlier work (Freuder & Wallace 1992; Shapiro & Haralick 1981) and for a conjunctive width heuristic that uses smallest domain size to break ties. First of all, note that P-EFC with dynamic search rearrangement outperforms P-EFC based on random (lexical) ordering (the best algorithm in our earlier work) by two orders of magnitude for larger problems. This is in contrast to the results of Shapiro and Haralick, who found only a slight improvement; however, their problems were more homogeneous and had complete constraint graphs. Results for P-BB and RPO are only given for 10- and 15-variable problems because of the difficulty in obtaining data for the full sample of larger problems. For P-BB, we estimate that it would take between 100 million and one billion constraint checks on average to solve the 20-variable problems.

In general, the conjunctive width algorithm outperforms all others. It requires 10^2 constraint checks on average to find an maximal solution for 10-variable problems, and for 20-variable problems its performance is still on the order of 10^4 constraint checks. In contrast, dynamic forward checking requires 10^5 constraint checks for larger problems. And this is far superior to the performance of the other algorithms. At 20 variables the range in performance covers 4-5 orders of magnitude.

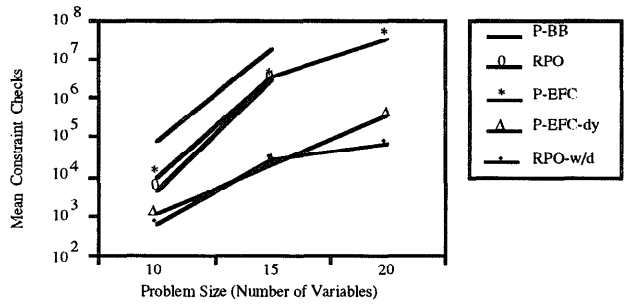

Figure 1. Performance of various algorithm on sparse random problems of varying size.

Table 1 shows results for RPO with conjunctive width heuristics based on two or three heuristic factors, for 15- and 20-variable problems. Dynamic P-EFC algorithms, some of which use value ordering based on ACCs are also shown. The most effective combinations of conjunctive heuristics used mean ACC as the first tie breaker, although at 20 variables domain size was almost as efficient. Degree, used in this capacity, produced results that were of the same order as dynamic P-EFC based on domain size, although this heuristic was sometimes effective as a second tie breaker.

Conjunctive width heuristics were also effective in combination with P-EFC, and the addition of value ordering based on ACCs led to further enhancement. The latter effect was not as large or consistent when value ordering was used with dynamic search order based on domain size; for one 20-variable problem performance was made dramatically worse, which accounted for most of the increase in the mean (cf. Table 1). As a consequence, for larger problems dynamic search rearrangement based on conjunctive width heuristics was markedly superior to the classical strategy of search rearrangement based on domain size.

Timing data was also collected for these heuristics (on a Macintosh SE/30; it was not possible to collect data for all of the weaker heuristics, but they were clearly inefficient overall). In general the pattern of results for this measure resembles that for constraint checks. However, the difference between RPO and P-EFC for a given width heuristic is generally greater than the difference in constraint checks, which indicates that the latter algorithms performed more background work per constraint check.

4 Factorial Analyses

In these new algorithms several factors work together to produce the impressive performance observed. In this section we will attempt to separate the effects of these factors experimentally.

The purpose of the first series of tests was to demonstrate that each of the separate factors did affect performance and to determine the effects of combining them. This required a factorial approach. To this end, P-BB and P-BMK were both tested in their basic form, then with either the domain value ordering based on ACCs or with ACCs used for lower bound calculations during search, and then with both factors. In addition, several ordering heuristics were tested in combination with the other factors. Finally, P-EFC with a fixed variable ordering was tested with the same variable order heuristics, as well as value ordering based on ACCs. These tests were done with the 10-variable problems since there were many conditions to run; in addition, basic features of these comparisons are shown more clearly with these smaller problems.

Table 1
Mean Constraint Checks (and Times) to Find a Maximal Solution for Conjunctive Width Heuristics

	Number of Variables (measure)		
	15	20	20
heuristics	(10^3 ccks)	(10^3 ccks)	(secs)
width/dom-sz	27.1	62.8	488
width/mean-ACC	5.7	46.0	362
width/dom-sz/mean-ACC	18.1	40.3	325
width/dom-sz/degree	21.6	36.5	293
width/mean-ACC/dom-sz	4.9	35.6	240
width/mean-ACC/degree	5.6	36.3	343
width/degree/dom-sz	36.5	493.7	--
width/degree/mean-ACC	42.5	330.2	--
dynamic P-EFC - dom-sz	20.9	336.9	3098
dynam. P-EFC - dom-sz (val)	16.7	722.4	--
dynam. P-EFC - wid/dom/AC	65.1	83.0	--
dyn. P-EFC -wid/dm/AC(val)	46.0	53.7	584
dynam. P-EFC -wid/AC/dom	28.6	187.1	--
dyn. P-EFC -wid/AC/dm (val)	7.0	38.9	274

Of the many interesting features of the results (see Tables 2a and 2b) these will be noted:

(i) each of the factors, variable ordering, value ordering and use of ACCs during search, has an effect on performance (compare entries along a row); moreover, these effects can be combined to produce more impressive performance than is obtained in isolation. This is true for P-BMK as well as the basic branch and bound (P-BB).

Table 2a
Factorial Analysis of Branch and Bound Variants
(Mean Constraint Checks in 1000s)

Variable Order	P-BB	BB/val	BB/ACC	BB/ACC/v
lexical	69.8	42.9	17.1	9.5
domsz	21.0	16.0	5.9	6.7
width	6.4	3.4	1.8	1.2
mean-ACC	31.6	12.2	20.1	2.3
dom/wid	14.6	12.7	4.7	4.3
dom/ACC	14.2	13.6	4.9	5.1
wid/dom	3.7	2.5	1.5	0.8
wid/ACC	2.5	1.1	1.8	0.7

Table 2b
Factorial Analysis of Backmark and P-EFC Variants
(Mean Constraint Checks in 1000s)

Var. Order	BM	BM/v	BM/AC	BM/AC/v	EFC	EFC/v
lexical	17.1	10.6	6.8	4.5	9.6	5.5
domsz	1.4	1.8	1.0	1.2	1.0	1.1
width	2.7	1.6	1.1	0.8	2.5	1.6
mean-ACC	9.6	4.5	6.3	0.8	1.0	0.6
dom/wid	1.4	1.5	0.9	0.8	0.7	0.8
dom/ACC	1.5	1.7	1.0	1.1	0.9	0.7
wid/dom	1.2	1.1	0.8	0.6	1.1	1.1
wid/ACC	0.7	0.6	0.8	0.5	0.8	0.6

(ii) For P-BB, width at a node is superior to the other variable ordering heuristics tested. In addition, conjunctive heuristics improve performance for both width and domain size, with the former retaining its superiority.

(iii) For P-BMK the superiority of width is not as apparent when constraint checks alone are considered. However, the means for the domain size heuristic do not reflect the larger number of nodes searched. (BMK avoids many constraint checks in this case with its table lookup techniques.) At the same time, conjunctive width heuristics are consistently superior to conjunctive domain heuristics.

(iv) Width and conjunctive width heuristics are more consistently enhanced by the combination of ACCs and

value ordering than other variable ordering heuristics. In this combination, they also usually outperform forward checking based on the same heuristic.

For larger problems, the trends observed with smaller problems were greatly exacerbated, and at 20-variables it was difficult to obtain data for all problems with RPO based on domain size or mean ACCs. (For both simple and conjunctive two-tiered heuristics based on domain size or mean ACC, the number of constraint checks and nodes searched reached eight or nine orders of magnitude for some problems.) For this reason, these heuristics will not be discussed further.

In addition, for larger problems the simple width heuristic is less effective overall (for 20-variable problems it was not appreciably better than dynamic forward checking), and it is here that conjunctive heuristics based on width afford significant improvement. There is also a minority of the problems of larger size that cannot be solved without an extraordinary amount of effort by either the 'pure algorithms' or by some of the partial combinations; however, at least one of the strategies that make up a combination is usually very effective. Both effects are observed for two difficult problems in the 20-variable group (Table 3; for problem #3, the width/domain-size/mean-ACC ordering gave the same results as width/domain-size). In both cases P-BMK based on width alone does poorly. For problem #3, either conjunctive ordering or use of ACCs has some effect, but not enough to make the problem easy; however, ordering domain values makes the problem trivial. In contrast, this strategy has almost no effect on problem #10 when used alone; conjunctive ordering and ACCs are both effective, and in combination with these strategies, value ordering affords further improvement, so that this problem also becomes relatively easy to solve.

Table 3
Factorial Analysis for Individual 20-Variable Problems

		Problem #3		Problem #10		
		width	wid/dom	width	wid/do	w/d/ ACC
BMK	BT	84E6	36E6	18E6	4E5	9E4
	CCK	159E6	2E6	62E6	1E6	3E5
/val	BT	4E2	2E2	18E6	4E5	4E4
	CCK	1E3	8E2	60E6	1E6	1E5
/ACC	BT	53E6	31E6	2E6	1E5	4E4
	CCK	45E6	2E6	4E6	3E5	9E4
RPO	BT	1E2	1E2	2E6	1E5	1E4
	CCK	2E2	3E2	4E6	2E5	2E4

Combination algorithms are therefore effective in part because they 'cover all the bases'. However, this does not explain the peculiar effectiveness of the conjunctive width heuristics, and there is still the question of how the different factors actually enhance branch and bound. These questions are treated in the next section.

5 Analysis of Individual Strategies

The analyses in this section assess the effects of different factors on the upper and lower bounds that are used during search, as well as interactions between factors with respect to these bounds. For ease of analysis and exposition this discussion is restricted to retrospective algorithms.

Value ordering based on ACCs should insure that good solutions are found sooner, which will reduce the upper bound more quickly. This is confirmed by the results in Figure 2, which also shows that this reduction is independent of either the variable orderings or the inclusion of ACCs in the lower bound. With this form of value ordering, a maximal solution is obtained by the fifth try in all but two of the 25 ten-variable problems. Similar results were found with larger problems.

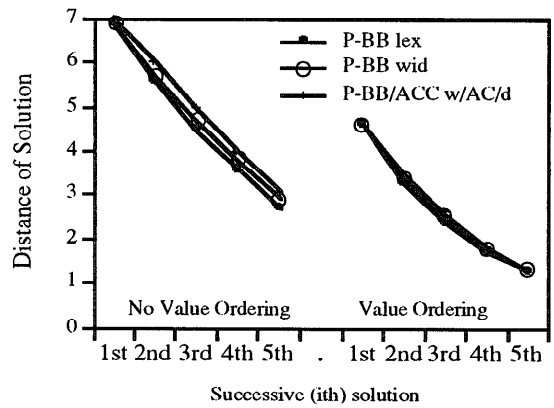

Figure 2. Mean distance (upper bound) after successive solutions found during search, with and without value ordering by increasing AC counts (10-variable problems). When a solution before the fifth was maximal, its distance was used in later group means.e 15-variable problems.

Variable ordering can influence lower bound calculations by affecting either the increase in distance (current cost) or the projected costs based on ACCs (or FC counts). To see this we will consider three orderings: lexical (a random ordering), ordering by selecting the variable with the highest mean ACC (a straightforward Fail First strategy), and the width/domain-size ordering. Figure 3 shows mean ACC as a function of position in the variable ordering for the 10-variable problems. This is the expected value to be added to the current distance in calculating the lower bound. As expected, lexical ordering shows no trend in this measure, while ordering by mean ACC shows a steep decline from the first position to the last. The width/domain-size ordering shows a sharp rise from the first to the second position, reflecting the fact that the smallest domains are almost completely supported;

thereafter, there are no obvious trends for the entire set of problems.

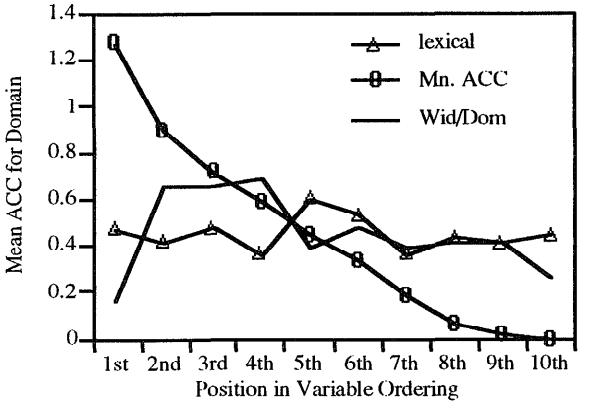

Figure 3. Mean ACCs at successive positions in variable ordering for three ordering heuristics, based on the 10-variable problems.

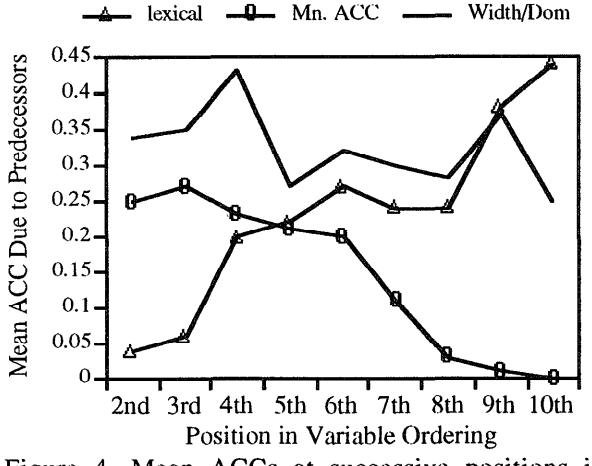

Figure 4. Mean ACCs at successive positions in variable ordering due to variables selected earlier in the search order for three ordering heuristics, based on the 10-variable problems.

Figure 4 shows a further analysis for the same problems, the mean ACC at each position due to variables that *precede* the one in question. This is the expected increase in distance that will be found during consistency checking for a value from the domain of this variable. The important points are, (i) for lexical ordering, ACCs are not translated into immediate increases in the distance until late in search, (ii) for mean ACC ordering, there is relatively little transformation of this sort (as a proportion of the mean ACC) and none late in search. To a great extent, this heuristic puts 'all its eggs in one basket' by maximizing the increase in the lower bound due to the ACC; if this strategy fails, search is likely to be very inefficient, (iii) width/domain-size ordering gives a relatively high rate of transformation of this sort throughout the search. Since the average ACC for this ordering heuristic is also relatively high throughout search, this heuristic tends to increase the lower bound through effects on both current distance and prospective components (giving a "one-two punch" via this bound).

These analyses indicate that conjunctive width orderings are effective when they place variables likely to produce inconsistencies and ACCs (e.g., those with small domain sizes) just ahead of variables that are adjacent and are, therefore, likely to have inconsistent values. This sets up the one-two punch effect described above.

6 Varying Parameter Values

The methods described in (Freuder & Wallace 1992) were used to generate random problems varying in number of arcs in the constraint graph as well as tightness of constraints and domain size. These methods generate problems for which the expected value of these parameters can be specified, and values tend to be normally distributed around the expectations. In this discussion, number of arcs will be described in terms of the average degree of a node in the constraint graph. All problems had 20 variables; the expected degree was 3, 4, 5 or 7; the expected domain size was either 2 or 4; and for each problem density (expected degree) there were three values of tightness that differed by 0.1, beginning with the smallest value for which it was relatively easy to get problems with no solutions. In the latter case, the mean distance for maximal solutions was 2-3; for the maximum tightness the mean distance was 7-12. RPO with width/domain-size/mean-ACC was compared with four versions of dynamic forward checking, using either search rearrangement based on domain size or on the width/domain-size/mean-ACC heuristic and with or without prior value ordering based on ACCs. (Note. With this method, it was possible to generate constraints with all possible pairs; this happened with some of the smaller values for tightness. However, there were never more than a few such constraints in any problem.)

For problems of expected degree 3 (equal to that of the initial set), RPO was always better than dynamic P-EFC based on domain size, usually by an order of magnitude (e.g., 7774 ccks on average vs. 66,035 for tightness = 0.7). Hence, the effectiveness of RPO does not depend on such pecularities of the first set of problems as inclusion of variables with very small domain sizes and small maximal distances. Problem difficulty increased with constraint tightness; for the most difficult problems RPO and dynamic P-EFC using width/domain-size/mean-ACC and value ordering had about the same efficiency (71,036 vs. 68,502 mean ccks, respectively, for tightness = 0.8).

Dynamic P-EFC based on domain size did better with respect to RPO as the density of the constraint graph increased; however, the point of crossover between them varied depending on constraint tightness. For problems of low tightness (easier problems) the crossover was between 4 and 5 degrees; for problems with tighter constraints it was around 5 degrees. There was evidence that dynamic P-

EFC based on width/domain-size/mean-ACC outperforms dynamic P-EFC based on domain size up to an average degree of 7 (e.g., for the tightest constraint at degree 5, mean ccks were 220,865 and 348,074, respectively, for these algorithms).

Thus, algorithms with conjunctive heuristics outperform all others on random problems with average degree ≤ 4. Within this range, forward checking tends to be the best algorithm for problems with higher density and greater average constraint tightness, and RPO is best for problems of lower density and looser constraints. As density (average degree) increases, dynamic P-EFC based on domain size eventually predominates; this occurs sooner for problems with looser constraints (which are easier problems for these algorithms). In contrast to Experiment 1 this standard algorithm was also improved when values were ordered according to ACCs.

7 Conclusions

Conjunctive width heuristics can enhance branch and bound algorithms based on either prospective or retrospective strategies. In combination with preprocessing techniques, the resulting algorithms outperform other branch and bound algorithms by at least one order of magnitude on a large class of problems. Maximal solutions can now be obtained for some problems of moderate size (20-30 variables, depending on specific parameter values) in reasonable times. Since branch and bound can return the best solution found so far at any time during search [Freuder & Wallace 92], these new algorithms may also perform well in relation to existing algorithms that find nonoptimal solutions (e.g., [Feldman & Golumbic 90]), for an even larger class of problems.

References

Dechter, R., and Meiri, I. 1989. Experimental evaluation of preprocessing techniques in constraint satisfaction problems, In *Proceedings IJCAI-89*, Detroit, MI, p. 271-277.

Feldman, R., and Golumbic, M.C. 1990. Optimization algorithms for student scheduling via constraint satisfiability. *Comput. J.*, 33: 356-364.

Freuder, E.C. 1982. A sufficient condition for backtrack-free search. *J. Assoc. Comput. Mach.*, 29: 24-32.

Freuder, E., and Wallace, R.J. 1992. Partial constraint satisfaction. *Artif. Intell.*, 58: 21-70.

Haralick, R.M., and Elliott, G.L. 1980. Increasing tree search efficiency for constraint satisfaction problems. *Artific. Intell.*, 14: 263-313.

Reingold, E.M., Nievergelt, J., and Deo, N. 1977. *Combinatorial Algorithms. Theory and Practice.* Englewood Cliffs, NJ: Prentice-Hall.

Shapiro, L., and Haralick, R. 1981. Structural descriptions and inexact matching. *IEEE Trans. Pattern Anal. Machine Intell.*, 3: 504-519