

Dynamic Constraint Satisfaction Problems: Relations between Search Strategies and Variability in Performance ^{*}

Richard J. Wallace, Diarmuid Grimes and Eugene C. Freuder

Cork Constraint Computation Centre and Department of Computer Science
University College Cork, Cork, Ireland
email: {r.wallace,d.grimes,e.freuder}@4c.ucc.ie

Abstract. Previously we presented a new approach to solving dynamic constraint satisfaction problems (DCSPs) based on features of the problem that remain stable after small- or moderate-sized change. We also showed that even small changes in a CSP can have profound effects on the search performance of ordinary heuristics. The present work extends this analysis. We show that despite a reduction in search effort, variability after change is still pronounced, as reflected in low correlation coefficients. This is still true when effects on fail-firstness are separated from promise effects. Moreover, such variability does not depend on similarity in solution sets, reflected in the Hamming distance of the solution closest to the one found for the original problem, although this correlates well with efficiency of the local changes algorithm. Thus, methods based on identifying sources of contention improve average performance without reducing variation in performance after perturbation of individual problems.

1 Introduction

A “dynamic constraint satisfaction problem”, or DCSP, is defined as a sequence of CSPs in which each problem in the sequence is produced from the previous problem by changes such as addition and/or deletion of constraints [VJ05]. Although several strategies have been proposed for handling DCSPs, there is still considerable scope for improvement, in particular, when problems are in the critical complexity region. For these problems, algorithms that attempt to repair the previous solution can be grossly inefficient, especially when they are based on complete search [WGF09].

In recent work we found that for hard CSPs, search performance (amount of effort) can change drastically even after small alterations that do not change the values of the basic problem parameters. At the same time, one feature that is not greatly affected is the set of variables that are the major sources of contention within a problem. It follows that information derived from assessment of these sources of contention should enhance performance even after the problem has been altered. This is what we have found [WGF09]. We also showed that for these problems, a standard DCSP algorithm like local changes is 2-3 orders of magnitude worse, depending on the search heuristic.

^{*} This work was supported by Science Foundation Ireland under Grant 05/IN/1886.

More specifically, we showed that a heuristic procedure that uses failures during iterated sampling (“random probing”) can perform effectively after problem change, *using information obtained before such changes* and thus avoiding the cost of further sampling. The result is a new approach to solving DCSPs based on a robust strategy for ordering variables rather than solution repair or finding robust solutions.

Here, we show that despite this success, predictability of performance across a set of DCSPs of similar character, as reflected in the correlations between original and altered problems, remains fairly low. From an analysis of search performance based on the Policy Framework of [BPW04] [BPW05], we find that problem perturbation affects measures of both promise and fail-firstness, although the former is more strongly affected. We also find that existence of similar solutions does not predict variation in performance, although it does for local changes. This indicates that, while average performance can be improved significantly by identifying major points of contention, this still allows for marked differences in performance after small perturbations.

2 Background Material

2.1 Definitions and notation

Following [DD88] and [Bes91], we define a dynamic constraint satisfaction problem (DCSP) as a sequence of static CSPs, where each successive CSP is the result of changes in the preceding one. As in earlier work, we consider DCSPs with specific sequence lengths, especially length 1, where “length” is the number of successively altered problems starting from the first alteration to the initial problem.

In our extended notation, $P_{ij}(k)$ refers to the k th member in the sequence for $DCSP_{ij}$, where i is the (arbitrary) number of the initial problem in a set of problems, and j denotes the j th DCSP generated from problem i . However, for DCSPs of length 1 a simpler ij notation is often more perspicuous; in this case P_{i-j} is the j th problem generating by perturbing base problem i .

2.2 Experimental methods

In this paper, we restrict our inquiry to the case of addition *and* deletion of k constraints from a base CSP. In this case, the number of constraints remains the same. In addition, changes are carried out so that additions and deletions do not overlap.

Because they allow for greater control, the present experiments were done with random problems. (The original results included scheduling problems having intensional constraints and strongly ordered domains.) Problems were generated in accordance with Model B [GMP⁺01]. The base problems had 50 variables, domain size 10, graph density 0.184 and graph tightness 0.369. Problems with these parameters have 225 constraints in their constraint graphs. Although they are in a critical complexity region, these problems are small enough that they can be readily solved with the algorithms used.

DCSP sequences were formed starting with 25 independently generated initial problems. In most experiments, three DCSPs of length 1 were used, starting from the same

base problem. Since the effects we observed are so strong, a sample of three was sufficient to show the effects of the particular changes we were interested in.

Search was done with two non-adaptive variable ordering heuristics: maximum forward degree (*fd*) and the FF2 heuristic of [SG98] (*ff2*). The latter chooses a variable that maximises the formula $(1 - (1 - p_2^m)^{d_i})^{m_i}$, where m_i is the current domain size of v_i , d_i the future degree of v_i , m is the original domain size, and p_2 is the original average tightness. In addition, adaptive heuristics based on weighted degree were tested, including *dom/wdeg*, *wdeg* [BHLS04], and a version of search using *dom/wdeg* that uses weights at the start of search obtained by “random probing” [GW07]. This method involves a number of short ‘probes’ of the search space where search is run to a fixed cutoff and variable selection is random. Constraint weights are updated in the normal way during probing, but the information is not used until complete search begins. These heuristics were employed in connection with the maintained arc consistency algorithm using AC-3 (MAC-3). The performance measure was search nodes.

The basic demonstrations of DCSP effects involved search for one solution. To avoid effects due to vagaries of value selection that might be expected if a single value ordering was used, these experiments involved repeated runs on individual problems, with values chosen randomly. The number of runs per problem was always 100. The individual datum for each problem is mean search nodes over the set of 100 runs.

3 Basic Results

3.1 Non-adaptive search heuristics

This section includes some previous results (in some cases extended) to set the stage for the present work. Figure 1 (taken from [WGF09]) shows the extent of variation that can occur after small alterations, in this case addition and deletion of five constraints.

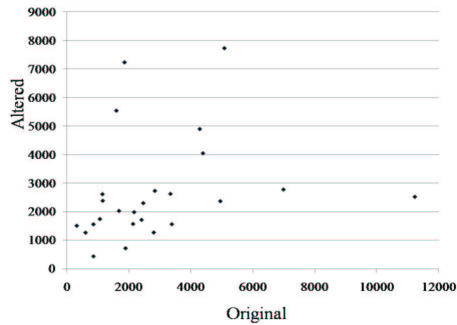


Fig. 1. Scatter plot of search effort (mean nodes over 100 runs) with *fd* on original versus $P_{i3}(1)$ problems with five constraints added and deleted. (Overall correlation in performance between the 25 original and altered problems is 0.24.)

Table 1 shows correlations for two sets of experiments (with the two different non-adaptive heuristics), where five constraints were added and deleted. Both here and in Figure 1, vagaries in search effort due to value ordering can be ruled out, since the statistics are based on means of 100 runs per problem with random value selection. In addition, our earlier work ruled out number of solutions as the major factor underlying such variation [WGF09]. These results show that even small changes can have a marked effect on search, and affect the relative difficulty of finding solutions in two problems before and after alteration.

Table 1. Correlations with Performance on Original Problems after Alteration: Non-Adaptive Heuristics

condit	<i>fd</i>			<i>ff2</i>		
	P1	P2	P3	P1	P2	P3
5c	.49	.83	.24	.34	.54	.31

Notes. <50,10,0.184,0.369> problems. Single solution search with repeated runs on each problem. “5c” is 5 deletions and additions. “Pj” = $P_{-j}(1)$.

3.2 Adaptive search heuristics

A major finding in previous work was that despite their marked effects on search, the changes just described often do not greatly affect the locations of major points of contention (bottleneck variables). Therefore, a heuristic that assesses major sources of contention should perform well, *even when using information from the base problem on a perturbed problem*. Since the constraint weights obtained from random probing distinguish points of high contention [GW07] [WG08], the usefulness of this information is not lost in the face of changes such as these. This is shown in Table 2 for the same problems used in Table 1. Data are for four methods: (i) *dom/wdeg* with no restarting, (ii) independent random probing for each problem (*randi*), (iii) a single phase of random probing on the original problems (*randi-orig*), after which these weights were used with the original and each of its altered problems (on each of the 100 runs with random value ordering), (iv) a strategy in which weights obtained from *dom/wdeg* on the base problem were used at the beginning of search with a perturbed problem. In the third and fourth cases, the new constraints in an altered problem were given an initial weight of 1. For these problems, random probing improves search performance in comparison with *dom/wdeg* and that there is relatively little fall-off if weights from the base problem are used. However, if weights from *dom/wdeg* are re-used, search performance is distinctly inferior to that found with weights obtained from probing. In fact, it is slightly worse than the original *dom/wdeg* heuristic, which starts search with no information other than degree. This shows the importance of gaining information about contention through random sampling of failures instead of in association with CSP search. It is also consistent with the proposal that random probing provides information about global sources of contention, in (partial) contrast to *dom/wdeg* [GW07].

Table 2. Search Results with Weighted Degree Heuristics

problems	<i>dom/wdeg</i>	<i>rndi</i>	<i>rndi-orig</i>	<i>d/wdg-orig</i>
random	1617	1170	1216	1764

Notes. Mean search nodes across all altered problems.

Table 3 shows correlations for the weighted degree strategies for the same DCSPs as in Table 1. Somewhat surprisingly, these correlations are very similar to the corresponding values found with non-adaptive heuristics. This shows that variability following problem perturbation does not decrease when these strategies are used. The variability is much the same degree when weighted degree is used in place of *dom/wdeg*. At the same time, variability is somewhat less, especially for P3, when the original weights from probing are used with the perturbed problems.

Table 3. Correlations with Performance on Original Problems after Alteration: Adaptive Heuristics

condit	P1 P2 P3			P1 P2 P3			P1 P2 P3		
	<i>d/wdeg</i>			<i>rndi-d/wdeg</i>			<i>rndi-orig-d/wdeg</i>		
5c	.49	.82	.26	.58	.76	.38	.59	.80	.43
	<i>wdeg</i>						<i>rndi-orig-wdeg</i>		
5c	.40	.82	.29				.61	.81	.60

Notes. See Table 1.

4 Changes in promise and fail-firstness

One hypothesis that could explain the improvement in search in spite of continuing variability for individual problems is that this is due to variability in promise. Previous work showed that contention strategies increase fail-firstness without greatly affecting promise [WG08].

Assessing promise and fail-firstness means measuring the adherence of a heuristic strategy to optimal choices under two conditions of search: (i) when search is on a solution path, i.e. the present partial assignment can be extended to a solution, (ii) when a mistake has been made and search is in an insoluble subtree. In the first case, an optimal policy would maximise the likelihood of remaining on the solution path; in the second, an optimal policy would minimise the size of the refutation (insoluble subtree) needed to prove the incorrectness of the initial wrong assignment. These policies are referred to as the “promise” and “fail-first” policies, and measures of adherence to each policy have been developed, which are referred to by the same names [BPW04] [BPW05].

The promise measure is basically a sum of probabilities across all complete search paths. Values can vary between 0 and 1, where a value of 1 means that any value in the domain will lead to a solution. The fail-first measure is the mean “mistake tree” size, where a mistake tree is an insoluble subtree rooted at the first non-viable assignment (i.e. the initial ‘mistake’). A larger mean mistake tree size therefore indicates poorer fail-firstness. In this paper, the analysis is based on an all-solutions search. To avoid artifacts due to averaging over different frequencies of mistakes at different search levels, comparisons for fail-firstness were restricted to individual levels of search.

For adaptive heuristics, there is the difficulty that promise and fail-firstness vary in the course of search. To avoid these effects, analysis was restricted to random probing with “frozen” weights, using the weights obtained from the base problems. This means that weights are not updated during search with the perturbed problems. If anything, this should elevate the correlations found. To remove the effect of varying domain size with the domain/degree strategy, random probing followed by simple weighted degree was also tested.

Table 4 (based on the same problems used in Table 1) gives a summary account of these differences in the form of correlations between each successive set of altered problems and the base problem set, for the non-adaptive heuristics. Again, lower correlations reflect changes in magnitude as well as differences in relative magnitude across an entire problem set. These data confirm the observation that greater variation occurs in connection with promise than with fail-firstness.

**Table 4. Correlations with Original Problems
(Non-adaptive Heuristics)**

measure	<i>fd</i>				<i>ff2</i>			
	P1	P2	P3	$\bar{x}(1-10)$	P1	P2	P3	$\bar{x}(1-10)$
prom	.22	.74	.34	.46	.50	.43	.16	.41
ff-1	.49	.67	.74	.69	.52	.47	.45	.58
ff-2	.75	.72	.72	.72	.77	.84	.46	.70

Notes. <50,10,0.184,0.369> problems. 5 constraints added and deleted. “prom” is promise measure. “ff-k” is mean size of mistake trees rooted at level *k*. Mean is for ten sets of DCSPs (only first three are shown).

The results in Table 5 show that using heuristics based on contention information can still yield low correlations on par with those obtained with non-adaptive heuristics. Removing the effect of domain size does not affect the overall pattern of results.

**Table 5. Correlations with Original Problems
(Adaptive Heuristics)**

measure	<i>rdi-orig-frz</i>				<i>rdi-orig-frz-wdeg</i>			
	P1	P2	P3	$\bar{x}(1-10)$	P1	P2	P3	$\bar{x}(1-10)$
prom	.82	.66	.22	.36	.39	.49	.28	.49
ff-1	.68	.73	.69	.57	.41	.57	.54	.62
ff-2	.86	.86	.78	.53	.79	.80	.65	.79

Notes. See Table 4.

If alterations of this sort have greater effects on promise than fail-firstness, then correlations for search effort before and after perturbation should be higher when problems have no solutions. This is, in fact, what is found. Table 6 shows results for three sets of perturbed problems; again, correlations are between performance on base problems and each of three sets of perturbed problems. Problems have similar parameters as those used in earlier tables, although the density was increased slightly to reduce the probability of generating problems with solutions.

Table 6. Correlations with Performance on Insoluble Problems after Alteration: Non-Adaptive Heuristics

condit	<i>fd</i>			<i>ff2</i>		
	P1	P2	P3	P1	P2	P3
5c	.80	.85	.84	.90	.91	.93

Notes. <50,10,0.19,0.369> problems. Both base and perturbed problems were insoluble. “5c” is 5 deletions and additions. “Pj” = $P_{-j}(1)$.

The contention-based heuristics also had consistently high correlations for these insoluble problems, as shown in Table 7. In terms of average nodes over the 75 perturbed problems, there was little fall-off between *rndi-orig* and *rndi* (3812 and 3778 resp.), while both improved over *dom/wdeg* (5268 nodes). These results match those found for the soluble problem set.

Correlations for search effort on insoluble problems, where performance differences are solely due to differences in fail-firstness, are appreciably higher than those for the fail-firstness measures in Tables 4-5. This may be due to greater sampling artifacts in the latter case, since sampling was limited to single levels of search. In addition, search performance is a cumulative rather than an intensity measure, since search effort is summed across several mistake trees.

Table 7. Correlations with Original Insoluble Problems: Adaptive Heuristics

heuristics	P1	P2	P3
<i>dom/wdeg</i>	.88	.91	.86
<i>rndi</i>	.84	.86	.89
<i>rndi-orig</i>	.90	.91	.86

Notes. See Table 6.

5 Nearest Solution Analysis

Earlier results showed that the number of solutions can change drastically after the kind of change we are considering [WGF09]. However, we also know that number of solutions is only weakly correlated with search effort. Another aspect of change pertains to the character of the solution set. If this changes appreciably, then expected search effort should also change. We can assess this aspect by calculating Hamming distances between solutions in the original and perturbed problems. Specifically, given a solution to the base problem, we can determine the Hamming distance of the *nearest* solution (minimal Hamming distance) in the perturbed problem. This can be done using a branch-and-bound search, where the number of differing assignments serves as the bound.

In these experiments, each of the 25 base problems was solved 100 times, using *dom/wdeg* with random value ordering. For each solution, the minimum Hamming

distance was determined. Here, Hamming distances can range between 0 (meaning the solution to the base was still a solution for the perturbed problem) and 50 (meaning no assignment in the solution to the base problem was part of any solution to the perturbed problem).

Figure 2 illustrates the extremes in results using three different perturbed problems. For clarity, Hamming distances were ordered from smallest to largest across the hundred runs for each problem. For problem P11-2, the minimal Hamming distance was always low, indicating that the solution set always contained similar solutions to the one found for the base problem; in contrast, for P6-2 the minimal Hamming distance was ≥ 40 . Other problems, such as P23-1 fell between these extremes; here, the minimal Hamming distance ranged from 0 to 38. Over the 75 P1, P2 and P3 problems, average minimal Hamming distance per problem over 100 runs ranged from 0.4 to 42.7.

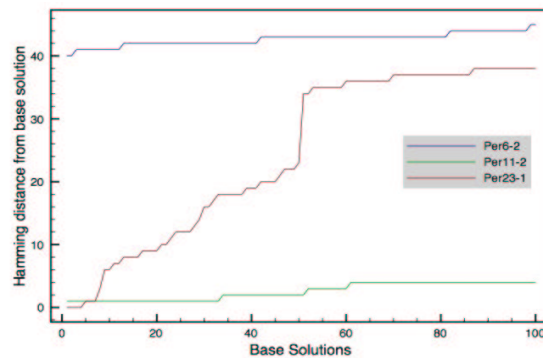


Fig. 2. Hamming distances of nearest solutions based on 100 different solutions to the corresponding base problem, for three different perturbed problems

For the three problems whose data are shown in Figure 2, weight profiles based on random probing on the original problem and on the perturbed problem were highly correlated (≥ 0.92). This is typical (see [WGF09]), and is, of course, the basis for the strategy of using weights from probes of the base problem with perturbed problems. At the same time, variation in performance after perturbation was found for all three problems, and was not related to the minimal Hamming distances. Local changes, on the other hand, showed differences that were closely related to the minimal Hamming distance. For example, for local changes with min-conflicts value ordering and $ff2$ for variable selection, nodes explored were 3, 159,091, and 216,914, for problems P11-2, P23-1, and P6-2, respectively. The same ordering was found using fd , although in this case search effort for the latter two problems was an order of magnitude greater.

6 Conclusions

In earlier work we presented a new approach for solving a range of DCSPs much more effectively than previous methods. We also presented a rationale for this strategy. However, as shown in the present paper, an apparent paradox arises when we perform the same correlational analysis as we did in earlier work with non-adaptive heuristics. This analysis shows that, in spite of using information about problem features that remain stable in the face of change, we still find marked variation in performance, reflected in modest or low correlations.

One hypothesis that we considered was that, since probing enhances fail-firstness but not promise, the variability was related to variability in promise. The present results show that problem alterations have effects on both promise and fail-firstness, although the effects are proportionally greater in the former case. However, adaptive contention-based strategies yield no improvement in predictability (reflected in similar correlations) for fail-firstness. So this hypothesis must be considered at best incomplete.

Moreover, we show that strategies that do not use previous solutions to guide search after problem alteration, cannot reduce variability even when there are similar solutions in the perturbed problem. At the same time, these strategies are more robust in the face of differences in the solution sets. As a result, they show much better overall performance following problem change than algorithms designed to take advantage of solution similarity.

References

- [Bes91] C. Bessière. Arc-consistency in dynamic constraint satisfaction problems. In *Proc. Ninth National Conference on Artificial Intelligence-AAAI'91*, pages 221–226. AAAI Press, 1991.
- [BHLS04] F. Boussemart, F. Hemery, C. Lecoutre, and L. Sais. Boosting systematic search by weighting constraints. In *Proc. Sixteenth European Conference on Artificial Intelligence-ECAI'04*, pages 146–150. IOS, 2004.
- [BPW04] J. C. Beck, P. Prosser, and R. J. Wallace. Variable ordering heuristics show promise. In *Principles and Practice of Constraint Programming-CP'04. LNCS No. 3258*, pages 711–715. Springer, 2004.
- [BPW05] J. C. Beck, P. Prosser, and R. J. Wallace. Trying again to fail-first. In B. Faltings, A. Petcu, F. Fages, and F. Rossi, editors, *Recent Advances in Constraints-CSCP 2004. LNAI No. 3419*, pages 41–55. Springer, 2005.
- [DD88] R. Dechter and A. Dechter. Belief maintenance in dynamic constraint networks. In *Proc. Seventh National Conference on Artificial Intelligence-AAAI'88*, pages 37–42. AAAI Press, 1988.
- [GMP⁺01] I. P. Gent, E. MacIntyre, P. Prosser, B. M. Smith, and T. Walsh. Random constraint satisfaction: Flaws and structure. *Constraints*, 6:345–372, 2001.
- [GW07] D. Grimes and R. J. Wallace. Learning to identify global bottlenecks in constraint satisfaction search. In *Twentieth International FLAIRS Conference*, pages 592–598. AAAI Press, 2007.
- [SG98] B. M. Smith and S. A. Grant. Trying harder to fail first. In *Proc. Thirteenth European Conference on Artificial Intelligence-ECAI'98*, pages 249–253. Wiley, 1998.
- [VJ05] G. Verfaillie and N. Jussien. Constraint solving in uncertain and dynamic environments: A survey. *Constraints*, 10(3):253–281, 2005.

- [WG08] R. J. Wallace and D. Grimes. Experimental studies of variable selection strategies based on constraint weights. *Journal of Algorithms: Algorithms in Cognition, Informatics and Logic*, 63:114–129, 2008.
- [WGF09] R. J. Wallace, D. Grimes, and E. C. Freuder. Solving dynamic constraint satisfaction problems by identifying stable features. In *Twenty-First International Joint Conference on Artificial Intelligence, IJCAI'09*, page to appear, 2009.