Dynamic Constraint Satisfaction Problems: Relations among Search Strategies, Solution Sets and Algorithm Performance*

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

Cork Constraint Computation Centre and Department of Computer Science
University College Cork, Cork, Ireland
{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 detection of major bottlenecks in a problem using a weighted-degree method called "random probing". The present work extends this approach and the analysis of the performance of this algorithm. We first show that despite a reduction in search effort, variability in search effort with random probing after problem perturbation is still pronounced, reflected in low correlations between performance measures on the original and perturbed problems. Using an analysis that separates effects based on promise and fail-firstness, we show that such variability is mostly due to variation in promise. Moreover, the stability of fail-firstness is greater when random probiing is used than with non-adaptive heuristics. We then present an enhancement of our original probing procedure, called "random probing with solution guidance", which improves average performance (as well as solution stability). Finally, we present an analysis of the nearest solution in the perturbed problem to the solution found for the original (base) problem. These results show why solution repair methods do poorly when problems are in a critical complexity region, since there may be no solutions similar to the original one in the perturbed problem. They also show that on average probing with solution guidance finds solutions with near-maximal stability under these conditions.

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

* This work was supported by Science Foundation Ireland under Grant 05/IN/I886. We thank E. Hebrard for contributing the experiment shown in Figure 2.

J. Larrosa and B. O'Sullivan (Eds.): CSCLP 2009, LNAI 6384, pp. 105–121, 2011.

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.

More specifically, we showed that a heuristic procedure that uses failures obtained 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 promise to a much greater extent than measures of fail-firstness, which largely explains the anomalous findings.

Based in part on these findings (but also with an eye toward improving solution stability), we developed an enhanced algorithm which uses information in the solution to guide *value* selection (called "random probing with solution guidance"). This allows us to gain the benefits obtained by solution repair techniques such as Local Changes; specifically, search can be strongly limited or avoided entirely when a solution identical to or very similar to the solution to the base problem is in the set of solutions to the perturbed problem.

We also present results of a nearest solution analysis, that is, a comparison of the solution found for the base problem with the closest solution among all solutions to the perturbed problem. This analysis shows that, for problems in the critical complexity region, often there are no similar solutions; this is why solution repair methods perform poorly on average on these problems. On the other hand, random probing with solution guidance can enhance performance whether or not closely similar solutions can be found in the perturbed problem.

The next section gives some background material, including descriptions of the problems used in this study. The third section presents performance data including mean performance and correlations between performance on the base problems and on the perturbed problems. The fourth section presents an analysis in terms of measures of adherence to the promise and fail-first policies. The fifth section describes the random probing with solution guidance procedure. The sixth section presents the nearest solution analysis. The last section gives conclusions.

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 kth 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 jth DCSP generated from problem i. However, for DCSPs of length 1 a simpler *ij* notation is often more perspicuous; in this case Pij is the jth problem (equivalent in this case to the jth DCSP) generated by perturbing base problem i. For mean values (shown in some tables), since i values range over the same set, this notation can be simplified further, to P-j or Pj.

2.2 Experimental Methods

Because they allow for greater control, many of the present experiments were done with random problems, generated in accordance with Model B [GMP+01]. The base problems had 50 variables, domain size 10, graph density 0.184 and constraint 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 (together with good heuristics).

For these problems, we restrict our inquiry to the case of addition *and* deletion of k constraints from a base CSP. (Other kinds of change are described in [WGF09].) In this case, the number of constraints remains the same. In addition, changes are carried out so that additions and deletions do not overlap.

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 latter 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 reported here is search nodes, although constraint checks and runtimes were also recorded.

Experiments on problems with ordered domains involved simplified scheduling problems, used in a recent CSP solver competition[1]. These were "os-taillard-4" problems, derived from the Taillard benchmarks [Tai93], with the time window set to the

[1] `http:/www.cril.univ-artois.fr/~lecoutre/benchmarks/benchmarks.html`

best-known value (os-taillard-4-100, solvable) or to 95% of the best-known value (os-taillard-4-95, insoluble). Each of these sets contained ten problems. For these problems, constraints prevent two operations that form part of the same job or require the same resource from overlapping. Specifically, they are disjunctive relations of the form, $(X_i + dur_i \ \leq \ X_j) \bigvee (X_j + dur_j \leq X_i)$, where X_k is the start-time and dur_k the duration of operation k. These problems had 16 variables, the domains were ranges of integers starting from 0, with 100-200 values in a domain, and all variables had the same degree. In this case, the non-adaptive heuristic used was minimum domain/forward degree.

Scheduling problems were perturbed by changing upper bounds of a random sample of domains. In the original problems, domains of the 4-100 problems are all ten units greater than the corresponding 4-95 problems. Perturbed problems were obtained by decreasing four domains of the 4-100 problems by ten units or by increasing six of the domains of the 4-95 problems by ten units. Perturbed problems were selected so that those generated from the 4-100 set remained solvable, while those generated from the 4-95 set remained insoluble.

For solvable problems, the basic demonstrations of DCSP effects were based on a search for one solution. To avoid effects due to vagaries of value selection that might be expected if a single value ordering was used, in these experiments repeated runs were performed on individual problems, with values chosen randomly. (For scheduling problems, value ordering was randomised by choosing either the highest or lowest remaining value in a domain at random.) For random problems, the number of runs per problem was always 100; for scheduling problems with solutions the number was 50. The individual performance datum for each problem is, therefore, mean search nodes over a set of 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.

Table 1 shows the grand means for search with fd and $ff2$ across all 75 of the altered CSPs (i.e. the altered problems from the three sets of 25 DCSPs).

Table 1. Search Performance on Altered Problems: Non-Adaptive Heuristics

	fd	$ff2$
5c	2601	3561

Notes. <50,10,0.184,0.369> problems. Single solution search with repeated runs on each problem. "5c" is 5 deletions and additions. Mean search nodes.

Table 2 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

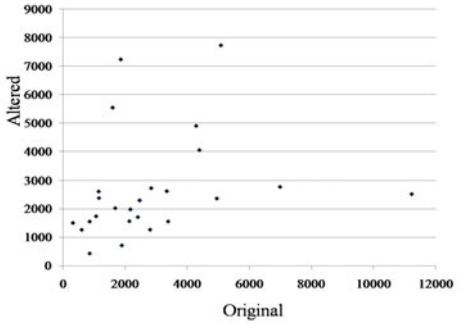

Fig. 1. Scatter plot of search effort (mean nodes over 100 runs) with fd on original versus $P_{i3}(1)$ problems with five constraints added and deleted. (Overall correlation in performance between the 25 original and altered problems is 0.24. Taken from [WGF09].)

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 problems before and after alteration.

Table 2. Correlations with Performance on Original Problems after Alteration: Non-Adaptive Heuristics

	fd			$ff2$		
condit	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 = $P_{-j}(1)$; thus, P1, P2 and P3 refer to three separate sets of DCSPs of length 1.

Table 3 shows correlations for os-taillard-4 problems for the non-adaptive heuristic, minimum domain over forward degree. Across five sets of perturbed problems, the mean number of search nodes was 257,695 for the 4-95 problems and 391,236 for the 4-100 problems.

Again, there are cases where small changes affect performance to such a degree that the correlations are negligible. That the correlations are often high is not surprising given the nature of the changes to the problems (and the global nature of the correlation coefficient). What is interesting is that even under these conditions changes in performance can occur that are sufficiently marked so that an ensemble measure of similarity can be affected. Moreover, it is possible to obtain marked differences (reflected in negligible correlation coefficients) even for problems without solutions.

Table 3. Correlations with Performance on Original Problems after Alteration for OS-Taillard Problems

			d/fd		
problems	P1	P2	P3	P4	P5
4-95	.03	.98	.99	1.00	.98
4-100	.51	.98	.99	.96	.13

Notes. Problems perturbed by incrementing or decrementing domains. Means for 4-100 problems based on 50 runs per problem with randomised value ordering as described under Methods.

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 4 for the same problems used in Tables 1 and 2. Data are for four methods (the first three of which were used in [WGF09]): (i) $dom/wdeg$ with no restarting, (ii) independent random probing for each problem ($rndi$), (iii) a single phase of random probing on the original problems ($rndi$-$orig$), 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 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 4. Search Results with Weighted Degree Heuristics: Random Problems

$dom/wdeg$	$rndi$	$rndi$-$orig$	d/wdg-$orig$
1617	1170	1216	1764

Notes. Mean search nodes across all altered problems. First three values from [WGF09].

Table 5 shows correlations for the weighted degree strategies for the same DCSPs as in Tables 1-2. 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 of the same degree when weighted degree ($wdeg$) is used in place of $dom/wdeg$, thus removing effects related to dynamic domain size. At the same time, variability is somewhat less when the original weights from probing are used with the perturbed problems.

A roughly similar situation obtains with the scheduling problems, as shown in Tables 6 and 7.

Table 5. Correlations with Performance on Original Problems after Alteration: Adaptive Heuristics

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

Notes. See Table 2.

Table 6. Search Results with Weighted Degree Heuristics: Scheduling Problems

problems	$dom/wdeg$	$rndi$	$rndi$-$orig$
taillard-4-95	16,745	4139	5198
taillard-4-100	11,340	7972	5999

Notes. Mean search nodes across all altered problems. Open shop scheduling problems. From [WGF09].

Table 7. Correlations with Performance on Original Problems after Alteration: Adaptive Heuristics

algorithm	P1	P2	P3	P4	P5
	os-taillard-4-95				
d/wdg	.90	.97	.89	.84	.89
$rndi$	.45	.37	.19	.70	.82
$rndi$-$orig$	.42	.24	.09	.09	.22
	os-taillard-4-100				
d/wdg	.15	.99	.99	.90	.97
$rndi$	.81	.51	.86	.81	.76
$rndi$-$orig$	1.00	.98	.99	.98	.86

Notes. Open shop scheduling problems. P1-P5 refer to five separate sets of DCSPs of length 1.

4 Changes in Promise and Fail-Firstness

The results in the last section leave us with a puzzle: random probing is effective in reducing search effort with perturbed problems, but we still find the same striking

variation in performance after small perturbations that we saw with non-adaptive heuristics, reflected in moderate to low correlations between the original and perturbed problems with respect to search effort.

A possible explanation for this discrepancy can be couched in terms of the recently developed Policy Framework of [BPW04, BPW05]. In this framework for backtrack search, search is considered to always be in one of two states: (i) it is on a solution path, i.e. the present partial assignment can be extended to a solution, (ii) a mistake has been made, and search is in an insoluble subtree. In each case, an *ideal* policy for making a decision can be characterised, i.e. a policy that would be optimal if it could actually be realised. 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. Although they cannot be realised in practice, they can be used to characterise good (or bad) performance of variable ordering heuristics. This is because there are measures of adherence to each policy that can be used to assess performance, which are referred to by the same names [BPW04, BPW05, Wal06].

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 any 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. To avoid artifacts that might arise because of variations in fail-firstness at different search depths, comparisons for fail-firstness were restricted to mistakes made at the same level of search.

It must be emphasized that these measures are not simply assessments of various features of search like mean depth of failure, backtracks at a given level of search, etc. Instead, they are genuine quality-of-search measures, just like total search nodes or run-time. This holds by virtue of their association with two forms of optimal decision making associated with the two conditions of search described above. The rationale for using these policy-based measures is that they give us a more articulated assessment of performance, based on a partition of the states of search into those that can lead to a solution and those that cannot. Put another way, the Policy Framework allows us to define quality-of-search measures specific to each of the two fundamentally distinct types of search-state.

Given this framework, a hypothesis that could explain the improvement in search in spite of continuing variability for individual problems is that the latter is due to variability in promise, i.e. in adherence to the promise policy. Random probing was, in fact, devised as a fail-first strategy [GW07], and previous work has shown that this strategy affects fail-firstness without greatly affecting promise [WG08].

In this work, we used a sampling strategy for assessing adherence to the fail-first policy that is superior to that used in earlier work [BPW05, Wal06]. In earlier work mean mistake tree size was found in connection with an all-solutions search; this meant that sample sizes were not equal either across problems or at different levels of search. Moreover, with this method, there is a confounding factor in that use of a given heuristic

above the level of the mistake may affect the efficiency of finding a refutation once a mistake has been made. In the present work, sampling was confined to mistakes at a single level of search, k. Variables and values were chosen at random down to this designated level of search. Below that, search was done with the heuristic being tested. Runs were discarded if a solution was found given the randomly chosen assignments. Otherwise, once search returned to the level of the mistake, the size of the mistake tree (i.e. of the refutation) could be calculated. In addition, for levels > 1, a further criterion was imposed. This is that there had to be a solution based on the original partial assignment for the first k-1 variables assigned. This condition ensures that the mistake at level k is the *first* mistake, and therefore is the true root of the insoluble subtree. If this condition is not met, this means that there is an invalid assignment above level k, and this will affect the average size of the mistake tree as well as introducing unwanted bias into the sampling procedure.

In the present work, the sample size was 100. Thus, for every problem and every level of the first mistake, data were collected for the same number of mistake-trees before obtaining a mean tree size. In addition, seven new sets of DCSPs were tested along with the three original sets.

Table 8. Correlations with Original Problems (Non-adaptive Heuristics)

	fd				$ff2$			
measure	P1	P2	P3	$\overline{x}$(1-10)	P1	P2	P3	$\overline{x}$(1-10)
prom	.22	.74	.34	.46	.50	.43	.16	.41
ff-1	.59	.83	.88	.78	.73	.72	.60	.68
ff-2	.67	.75	.87	.82	.58	.63	.72	.75
ff-3	.82	.80	.77	.81	.70	.83	.80	.78

Notes. <50,10,0.184,0.369> problems. 5 constraints added and deleted. "prom" is promise measure. "ff-k" referes to mistake trees rooted at level k. Mean is for ten sets of DCSPs (the three original sets are also shown individually).

For adaptive heuristics, there is the additional problem 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. Admittedly, this may elevate the correlations found.

Table 8 (based on the same problems used in Tables 1-2) 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, maximum forward degree and FF2. Again, lower correlations reflect changes in magnitude as well as differences in relative magnitude across an entire problem set. These data show that much greater variation occurs in connection with promise than with fail-firstness.

The results in Table 9 show that using heuristics based on contention information still results in low correlations for promise, while correlations for fail-firstness are higher than those found for non-adaptive heuristics.

Table 9. Correlations with Original Problems (Adaptive Heuristic)

	$rndi\text{-}orig\text{-}frz$			
measure	P1	P2	P3	$\overline{x}$(1-10)
prom	.82	.66	.22	.36
ff-1	.81	.88	.92	.90
ff-2	.82	.83	.93	.87
ff-3	.79	.87	.86	.86

Notes. See Table 8.

If alterations of this sort have greater effects on promise than fail-firstness, then correlations for search effort before and after perturbation should be also be high when problems have no solutions. (In this case, the only policy in force is the fail-first policy.) This is, in fact, what is found. Table 10 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 parameters similar to those used in earlier tables, although the density was increased slightly to reduce the probability of generating problems with solutions.

The weighted-degree heuristics also had consistently high correlations for these insoluble problems. In terms of average nodes over the 75 perturbed problems, there was little fall-off between $rndi\text{-}orig$ and $rndi$ (3812 and 3778 resp.), while both improved over $dom/wdeg$ (5268 nodes). These results match those found for the soluble problem set.

Unfortunately, it has not been possible to obtain similar kinds of data for the scheduling problems, using a non-adaptive heuristic like $dom/fwddegree$. This is because of the large number of solutions (making promise calculations difficult) and the large size of the mistake trees for some problems (making fail-firstness calculations difficult).

Table 10. Correlations for Insoluble Problems: Non-Adaptive and Adaptive Heuristics

heuristics	P1	P2	P3
fd	.80	.85	.84
$ff2$	.90	.91	.93
$dom/wdeg$	.88	.91	.86
$rndi$	.84	.86	.89
$rndi\text{-}orig$	.90	.91	.86

Notes. <50,10,0.19,0.369> problems. Both base and perturbed problems were insoluble. Perturbations were adding and deleting 5 constraints. "Pj" = $P_{-j}(1)$.

Table 11 gives data on the mean size of insoluble sub-trees for different heuristics for each of the first three levels of search, i.e. when mistakes are made at levels 1, 2, and 3. As one would expect, the size decreases quickly with increasing mistake-depth. In addition, differences in mean size among heuristics correspond to differences in total search nodes found in ordinary search (cf. Tables 1 and 4), although they are higher than values that would be obtained if the first k variables were selected by the same heuristic.

Table 11. Mean Size of Mistake Tree for Each of the First Three Levels of Search

level	fd	$ff2$	$rndi\text{-}orig$
1	2448	3575	1241
2	749	978	394
3	249	312	142

Notes. Means over 275 problems, 100 subtrees per problem at each depth.

5 An Enhanced Probing Procedure

Since variability is already small for fail-firstness, this suggests that further improvements in this direction will be difficult. On the other hand, we do not know of any variable ordering methods that improve promise specifically, since in most cases an ordering that enhances promise also enhances fail-firstness [Wal06]. However, promise can be improved by value orderings as well as by variable orderings (and in fact the concept was only extended to the latter recently [BPW04]).

In this connection, an obvious approach is to piggy-back a solution guidance strategy onto the basic probing procedure. Specifically, we should start testing values in a domain by using the original assignment to that variable. This will allow us to catch those cases where the original solution is still valid, or where there is a solution that is very similar to the original one. We tried this, and in addition we enhanced this strategy with a novel value ordering heuristic, used after the initial value selection. This was to choose as the next candidate assignment the one that was consistent with the highest number of original assignments in the unassigned variables. We call this method "probing with solution guidance". (Note that it is the search that is solution-guided, not the probing. Guiding the probing procedure in this way would, of course, make no sense since it would undermine the sampling strategy.)

With this enhancement mean search effort was reduced significantly, both for $dom/wdeg$ and for random probing. For $dom/wdeg$ the mean search nodes across 75 perturbed problems (P1-P3) was 954. For $rndi\text{-}orig$ the mean was 701. These are grand means, since the measure of search effort for each problem was the mean of 100 runs in which the base was solved using random value ordering, and the solution obtained was used in the manner described above with the perturbed problem. The reduction in search effort found here was due to the fact that search was greatly abbreviated if the original solution was still valid or if a solution was available that was close to the original with respect to number of common assignments. This also resulted in a smaller mean difference between $dom/wdeg$ and $rndi\text{-}orig$.

This new method has the added benefit of improving solution stability (the degree of similarity between the solution to the original problem and the solution to the altered problem). In fact, as shown elsewhere, with this new method there is a small but definite improvement over Local Changes, an algorithm designed to minimise the difference between the old solution and the one found after problem change [VS94].

6 Nearest Solution Analysis

In trying to understand the performance of DCSP algorithms, an important aspect of the problem is the changed solution set after perturbation. Evaluations of this sort may help us understand why solution repair techniques such as Local Changes perform poorly when problems are in the critical complexity region, in contrast to contention-based methods.

Earlier results showed that the number of solutions can change drastically after the kind of change we are considering [WGF09]. However, we also found that number of solutions is only weakly correlated with search effort.

Another aspect of change pertains to the elements in the solution set. If these change appreciably, then expected search effort should also change. Following [VS94], we can assess such change by calculating the Hamming distances between the solution found for the original problem and solutions in the perturbed problem. In particular, 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 limited discrepancy search (Figure 2) or with branch-and-bound search, where the number of differing assignments serves as the bound. For problems with 50 variables, Hamming distances can range from 0 (meaning the solution found for the base problem is still a solution for the perturbed problem) to 50 (meaning no assignment in the solution to the base problem is part of *any* solution for the perturbed problem).

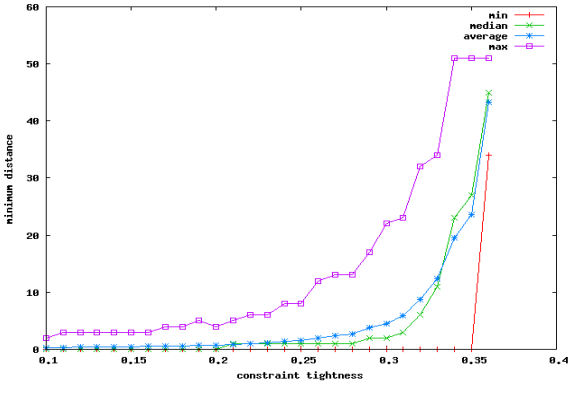

Fig. 2. Distance between the solution to the original problem and the nearest solution in the perturbed problem. Points are based on 100 or 500 randomly generated problems of 50 variables having the given tightness value.

Note that here we are not interested in similarity of the search path, but similarity between solutions. Nor do we need to consider the relation of the changes in assignments to the constraint graph, since, given a solution all constraints are satisfied. Our only concern is the amount of change necessary to employ the new solution rather than the original one, which only involves altering assignments. The amount of such change is, of course, directly measured by the Hamming distance.

Since the problems we have dealt with so far are in the critical complexity region, we first wanted to know how this affected the minimal Hamming distance. In these experiments, problems with the same basic parameters were used, except that constraint tightness was varied. (In addition, only three constraints were added and deleted in each perturbation.) $dom/wdeg$ was used to find a solution to the original problem generated. The most important result is that once one enters the critical complexity region, the minimal Hamming distance rises sharply, and the average approaches the maximum possible (Figure 2).

In subsequent experiments, we used the 5c problems discussed in Section 3. Each of the 25 base problems used previously was solved 100 times using $dom/wdeg$ with random value ordering. For each solution, the minimum Hamming distance was determined for one of the 75 perturbed problems. The average, minimum, maximum and median (minimal) Hamming distance for each problem are shown in Figure 3. For clarity, problems are ordered according to their average minimal Hamming distance over the 100 runs. For the 75 perturbed problems, the average minimal Hamming distance (over 100 runs) varied between 0.4 and 42.7, with a grand mean of 21.2. For 57 of the perturbed problems, there was at least one run where a solution was obtained for the base problem that gave a minimal Hamming distance of 0 when compared with all solutions to the perturbed problem. On the other hand, only 12% of the 7500 runs gave minimal Hamming distances of 0 for this comparison.

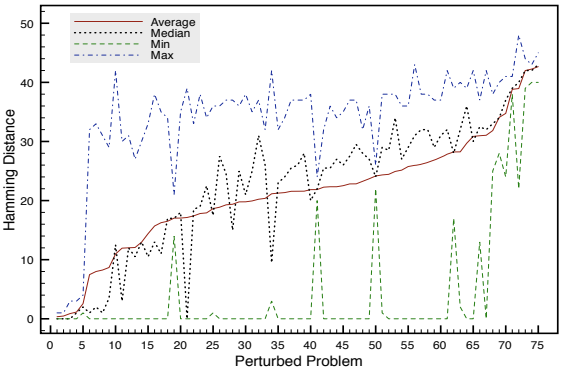

Fig. 3. Hamming distances between solution found for the base problem and closest solution in perturbed problem. Graphs show average minimal distance (solid line) together with the minimum, maximum and median value for each perturbed problem, based on 100 test runs.

Additional insight into these results can be obtained from complete distribution data for individual problems. Figure 4 illustrates the variation in such data using three perturbed problems. For clarity, Hamming distances were ordered from smallest to largest across the hundred runs for each problem. For problem P11-2 (the eleventh problem in the second set of DCSPs), 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 always ≥ 40.

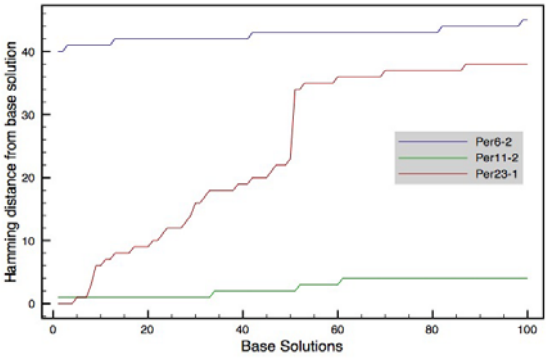

Fig. 4. Hamming distances of nearest solutions based on 100 different solutions to the corresponding base problem, for three different perturbed problems

Most problems fell between these extremes. An example, P23-1, is shown in the figure; here,the minimal Hamming distance ranged from 0 to 38.

These results give a clear indication of why solution reuse methods are often inefficient with hard problems. This is because when problems are hard, often none of the solutions in the perturbed problem are sufficiently close to the original solution for complete methods to gain from solution reuse.

For the three problems whose data are shown in Figure 4, the "weight profiles" (the summed weights of adjacent constraints for each variable) obtained after 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.

On the other hand, a solution reuse procedure like Local Changes showed differences that were related to the minimal Hamming distance. For example, for Local Changes with min-conflicts (look-back) value ordering and $ff2$ for variable selection, nodes explored were 3, 374,495, and 457,750, 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 one or two orders of magnitude greater.

As might be expected, for probing with solution guidance there is also a clear relation between the nearest solution and performance. This can be seen in Figure 5, which shows the results for individual runs for the three problems, in increasing order of search nodes. (For P11-2, the number of search nodes is always the minimum value of 50, although the minimal Hamming distance is not always 0. This probably reflects the winnowing effects of constraint propagation.)

In this case, these relationships are *not* reflected in a global measure like the correlation coefficient. This is because, as noted above, most problems show a large range for the minimal Hamming distance, depending on the solution found for the base problem. Thus, for Local Changes the correlation between mean minimal Hamming distance and

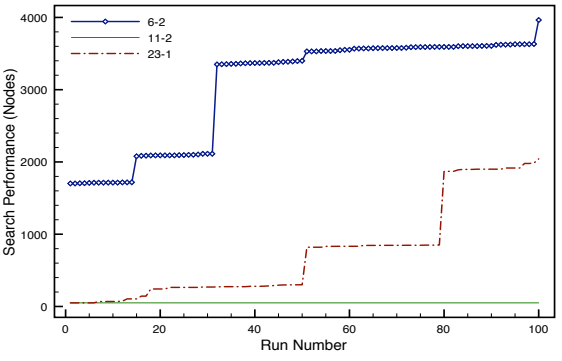

Fig. 5. Performance of probing with solution guidance with base-problem weights across 100 runs for the three perturbed problems in Figure 4

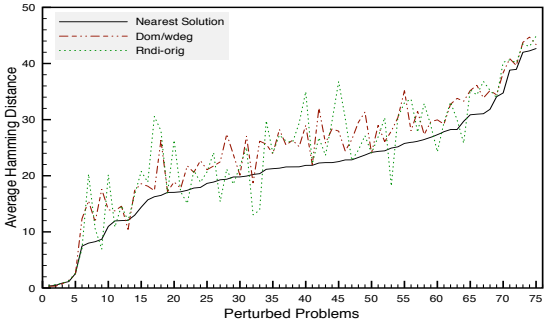

Fig. 6. Solution stability of adaptive heuristics using search with solution guidance, in comparison with average minimal Hamming distance from the earlier tests shown in Figure 3

search performance was 0.24, which is negligible. Interestingly, for $rndi\text{-}orig$ with solution guidance, the correlation was 0.53, still small but appreciably greater than that for Local Changes.

A final analysis shows that for probing with solution guidance, stability is in fact close to optimal. Figure 6 compares assessments of solution stability using the solution-based value ordering heuristic. Assessments were made by calculating the Hamming distance between the solution found for the original problem and that found for the perturbed problem; this was done 100 times for each perturbed problem. These averages are compared with the average minimal Hamming distance found for the same problems in the previous set of tests (cf. Figure 3). In the majority of cases, the means are comparable for each problem. (Naturally, the means based on single solution comparisons are usually higher than the corresponding mean minimal distances.) This indicates that with this method stability is close to the best value possible for these problems.

7 Summary and Conclusions

In earlier work we presented a new approach for solving a range of DCSPs that was much more effective 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.

In this paper, we consider the hypothesis that, since random probing enhances fail-firstness but not promise, the variability is related to variability in promise. Our results show that this hypothesis is largely correct. Of additional interest is the demonstration that random probing methods give greater predictability in performance with respect to fail-firstness than do non-adaptive heuristics. Here, we would also point out that this analysis demonstrates the usefulness of the Policy Framework for generating *and testing* hypotheses about heuristic performance.

This analysis was also useful in showing us where to look for improvements in performance – by showing us where not to look. In other words, since the issue involved promise rather than fail-firstness, this led us to consider strategies related to the former policy. Guided by this insight, we were able to devise a value ordering strategy which gives a large portion of the benefits of solution repair strategies such as Local Changes (with respect to performance and solution stability), without giving up any of the benefit of our basic contention-based strategy. Thus, mean search effort was reduced because search was avoided in those cases where the old solution was still viable or there was a viable solution that was very close to the original one.

This work also extends the analysis of the characteristics of DCSPs, a hitherto neglected area. We show that for difficult problems the solution set can change dramatically. As a result, it is often the case that there is no solution to the perturbed problem that is similar to the solution found for the original problem. This provides an explanation of why solution repair methods do not fare well for hard problems. This also suggests that the present methods, which rely on assessing the 'deep structure' of the problem, will probably be more robust in general than solution reuse methods.

References

[Bes91] Bessiére, C.: Arc-consistency in dynamic constraint satisfaction problems. In: Proc. Ninth National Conference on Artificial Intelligence, AAAI 1991, pp. 221–226. AAAI Press, Menlo Park (1991)

[BHLS04] Boussemart, F., Hemery, F., Lecoutre, C., Sais, L.: Boosting systematic search by weighting constraints. In: Proc. Sixteenth European Conference on Artificial Intelligence, ECAI 2004, pp. 146–150. IOS, Amsterdam (2004)

[BPW04] Beck, J.C., Prosser, P., Wallace, R.J.: Variable ordering heuristics show promise. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 711–715. Springer, Heidelberg (2004)

[BPW05] Beck, J.C., Prosser, P., Wallace, R.J.: Trying again to fail-first. In: Faltings, B., Petcu, A., Fages, F., Rossi, F. (eds.) CSCLP 2004. LNCS (LNAI), vol. 3419, pp. 41–55. Springer, Heidelberg (2005)

[DD88] Dechter, R., Dechter, A.: Belief maintenance in dynamic constraint networks. In: Proc. Seventh National Conference on Artificial Intelligence, AAAI 1988, pp. 37–42. AAAI Press, Menlo Park (1988)

[GMP+01] Gent, I.P., MacIntyre, E., Prosser, P., Smith, B.M., Walsh, T.: Random constraint satisfaction: Flaws and structure. Constraints 6, 345–372 (2001)

[GW07] Grimes, D., Wallace, R.J.: Learning to identify global bottlenecks in constraint satisfaction search. In: Twentieth International FLAIRS Conference, pp. 592–598. AAAI Press, Menlo Park (2007)

[SG98] Smith, B.M., Grant, S.A.: Trying harder to fail first. In: Proc. Thirteenth European Conference on Artificial Intelligence, ECAI 1998, pp. 249–253. Wiley, Chichester (1998)

[Tai93] Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 278–285 (1993)

[VJ05] Verfaillie, G., Jussien, N.: Constraint solving in uncertain and dynamic environments: A survey. Constraints 10(3), 253–281 (2005)

[VS94] Verfaillie, G., Schiex, T.: Solution reuse in dynamic constraint satisfaction problems. In: Twelth National Conference on Artificial Intelligence, AAAI 1994, pp. 307–312. AAAI Press, Menlo Park (1994)

[Wal06] Wallace, R.J.: Heuristic policy analysis and efficiency assessment in constraint satisfaction search. In: Proc. Eighteenth International Conference on Tools with Artificial Intelligence, ICTAI 2006, pp. 305–312. IEEE Press, Los Alamitos (2006)

[WG08] Wallace, R.J., Grimes, D.: Experimental studies of variable selection strategies based on constraint weights. Journal of Algorithms: Algorithms in Cognition, Informatics and Logic 63, 114–129 (2008)

[WGF09] Wallace, R.J., Grimes, D., Freuder, E.C.: Solving dynamic constraint satisfaction problems by identifying stable features. In: Twenty-First International Joint Conference on Artificial Intelligence, IJCAI 2009, pp. 621–627. AAAI/MIT (2009)