Toward Understanding Variable Ordering Heuristics for Constraint Satisfaction Problems

J. Christopher Beck[1] and **Patrick Prosser**[2] and **Richard J. Wallace**[1]

Abstract. Most previous work on understanding variable ordering heuristics for constraint satisfaction problems has focused on the ability to recover from bad decisions. It has been demonstrated however that this ability cannot be the full explanation of the quality of a variable ordering heuristic [8]. In this paper, we develop a more complete framework for analyzing heuristics based on optimal policies. We consider a second policy (in addition to the principle of quick recovery), which we call *promise*, which is simply to make choices that maximize the likelihood of successful search. We then develop a method for measuring the degree to which a heuristic conforms to the promise principle. Using this measure, we show that variable ordering heuristics vary with respect to their promise, and that for problems with many solutions the degree of promise correlates highly with efficiency of search.

1 INTRODUCTION

When presented with a number of choices, intelligence dictates that we select the best one according to some reasonable criterion. But how do we go about making a good decision? If we are on the path to reaching our goal, and we are presented with a number of options, a good decision will bring us closer to our goal. But if we have made a bad decision unknowingly and are no longer on a path to our goal, then we want to make decisions that will show us this as quickly as possible, so we can discover the bad decision and choose again. This means that in solving hard problems incrementally, we must deal with two different situations, where the assessment of decision quality may depend on different criteria.

Constraint satisfaction problems (CSPs) involve assigning values to variables in order to satisfy constraints among them. In this case, choices made during problem solving involve selecting which variable to consider next or which value to assign to this variable. Typically, some rule or *heuristic* is used to guide decision making at each step. For the most part, these rules are applied on a more-or-less ad hoc basis, and the reasons for the sometimes dramatic improvements in search efficiency are not yet clear. One criterion has been suggested for assessing heuristics, the well-known *fail first* principle. However, it has been demonstrated that adherence to this principle alone can actually impede search [8].

The main contention of this paper is that to understand heuristic performance, we must distinguish the two situations described above. We propose that each situation can be related to a basic principle or policy that defines what a good decision is, and that heuristics must be evaluated in relation to both principles. We call these principles *promise* and *fail-firstness*. *Promise* is the ability to make choices that lead to a solution when one exists, while *fail-firstness* is the ability to detect a wrong decision as soon as possible.

This paper focuses on the promise principle and on heuristics for variable selection. Although some variable ordering heuristics are known to enhance fail-firstness [6, 8], there is as yet no demonstration that they also vary with respect to promise. (On the other hand, it is expected that performance differences for value ordering heuristics will be due solely to differences in promise, although the size of these differences is unknown.) We first show how promise can be measured, which allows us to move from an abstract discussion of principles to concrete assessment of heuristics in these terms. We demonstrate that different heuristics do exhibit different degrees of promise and that the level of promise of a heuristic is correlated with its quality as measured by search cost. By taking performance principles like promise into account in evaluating heuristics, we may be able to clarify their effect on search performance. This may lead to more intelligent selection of heuristics and even to intelligent heuristic design.

1.1 Constraint satisfaction problems and search

In a constraint satisfaction problem (CSP) there is a set of variables, each with a domain of values, and a set of constraints acting between variables to restrict the set of possible value assignments. The problem is to find an assignment of values to variables that satisfies the constraints, or to prove that no such instantiation exists [9]. Many real world problems can be modeled as CSPs: scheduling problems, problems of design, routing, workforce management, etc. Typically a CSP is solved using a complete backtracking search. That is, a variable is selected for instantiation (the *current* variable) and is assigned a value. The un-assigned (*future*) variables are then filtered, removing all values from their domains that can be proved to be inconsistent with the past variable assignments. If all future variables have non-empty domains, we select from the future a new current variable. Otherwise we re-instantiate the current variable, or backtrack. The order that we select the current variable can have a profound effect on search effort [6, 7, 5]. We can use dynamic or static variable ordering heuristics. In a static variable ordering (SVO) heuristic variables are ordered before search starts, and are then always selected in that order. In contrast, dynamic variable ordering (DVO) heuristics select the current variable using information that is made available during the search process. Another primary component of search is the algorithm used to infer inconsistent values in the domains of future variables. A common consistency enforcement algorithm called forward checking [6], checks all values in the domain of the future

[1] Cork Constraint Computation Center Computing Science Department, University College Cork, Ireland {c.beck,r.wallace}@4c.ucc.ie
[2] Department of Computing Science, University of Glasgow, Scotland, pat@dcs.gla.ac.uk

variables against the assignment made to the current variable. If a value is inconsistent, it is removed from the domain.

Haralick & Elliott [6] proposed the *fail first* principle as a heuristic: *first try the places most likely to fail*. This was realized as SDF: the "smallest domain first" DVO heuristic which selects the variable with fewest values remaining in its domain. The justification for this was that it would reduce the average path length in the search tree, and this in turn would reduce search effort. Nudel [7] suggested another motivation for a heuristic, namely that it should minimize the number of nodes within the search tree. He presented an analytical model of the forward checking search process and showed that SDF will minimize the expected size of the search tree. In [8] Smith & Grant took the fail first principle to an extreme and engineered a heuristic that aggressively attempts to fail early in the search. While their heuristic did indeed reduce the average path lengths in the search tree, it did so with an increase in search cost. Clearly, the fail-first principle is not a full explanation for the quality of variable ordering heuristics.

2 RULES FOR DECISIONS: POLICIES AND HEURISTICS

When decisions are made during the course of search, it is helpful to have rules to guide the selections. Rules of this sort seem to have two basic forms, that have been called policies and heuristics, and that can be distinguished by the functions they perform. A *policy* identifies goals or end-results that are desirable. A *heuristic* identifies features of the situation that serve to distinguish among choices, such that a selection in these terms increases the likelihood of achieving the desirable goals.

It is significant that researchers in other disciplines have also seen fit to make this distinction. The following quotation is from an article in behavioral ecology (the study of behavior as adaptation) [3]:

> A policy model in our terminology is one which prescribes a general rule for maximizing payoff (the goal), without identifying any specific procedure which would enable the animal to follow the rule. (A policy for maximizing the goal of offspring production might, for example, be "Cross roads in a way that minimizes the chance of being run over", and the procedure for doing this might be "Look in both directions and cross if clear").

These authors talk about procedures or algorithms rather than heuristics, but the latter clearly form a part of any procedure in which nondeterministic choices must be made. Other workers in this field have, in fact, spoken of animals as following certain "rules-of-thumb" that allow them to approximate optimal policies.

There is an overall policy for search problems, which is to maximize the efficiency of the procedure, i.e. to minimize the effort required to find a solution, according to some relevant measure. An important measure of this sort is the number of nodes in the search tree, which is equal to the total number of assignments considered. Other measures, such as number of constraint checks and run time, attempt to better measure overall effort.

As indicated in the Introduction, two subordinate policies can be distinguished in this domain. This is because, for NP-complete and other hard combinatorial problems, search cannot be expected to make the best choice in all cases. Search may, therefore, enter a state where the partial solution cannot be extended to a complete one, i.e. the subtree below it does not contain any solutions. In this case, in order to succeed as quickly as possible, search should fail as quickly as possible so that it can get out of this subtree and return to a path that leads to a solution. This is in direct contrast to the situation where one is on such a path; in this case success should be followed by future success, to as great a degree as possible.

Promise has always been accepted implicitly as a policy in constraint solving, one that was probably too obvious to be described as such. On the other hand, the fail first principle is not so obvious, so its identification was (rightly) perceived as a discovery or a new proposal, and it has taken its place as a discernible strand in the lore of the field. Moreover, this proposal was first made in the context of the all-solutions problem, where for many algorithms the promise of different heuristics is the same. (Roughly, this is because all currently viable values in a domain must be tested in order to find all solutions.) As a result, the promise principle has tended to be disregarded. A further result is that the proper context of fail first has often been overlooked.

Another reason for the neglect of promise as a 'complementary' principle is that it is typically associated with value selection. In fact, an excellent value ordering heuristic is called the "promise" heuristic [4]. Value selection in turn is not usually expected to support fail-firstness. On this basis, one might hypothesize that while value ordering heuristics should support promise, variable ordering heuristics should support fail-firstness. In this case, one might question whether the distinction between policies and heuristics is important in practice, or even in theory. On the other hand, if variable ordering heuristics can also vary with respect to promise, then there can be no argument about the importance of this distinction, or the potential usefulness of considering heuristics in terms of how well they conform to basic policies.

3 A MEASURE OF PROMISE

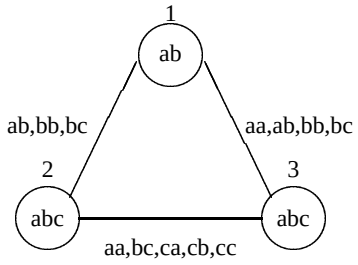
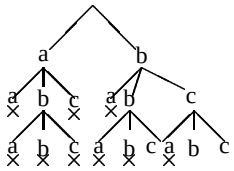


Figure 1. Example CSP with three variables, showing domains and the set of tuples in each constraint.

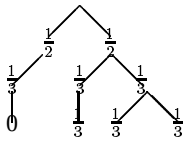
The promise policy is simply to make selections that have the greatest likelihood of succeeding. Because in practice we do not know which selection will maximize promise, we are forced to use heuristics. "Likelihood of success" can be treated probabilistically. That is, for a given decision there is some probability over all possible subsequent decisions that the choice will lead to success. We can also consider promise with respect to an entire problem, i.e. as an expected value over all possible sequences of choices. In this sense, we can speak of the promise of a problem in terms of its relation to a perfect selection. This measure of promise also has a natural minimum and maximum: 0 (for insoluble problems) and 1 (if every n-tuple is a solution). This gives us a universal measure (across all problems) that is linked directly to an optimal policy; if we can obtain a value for this measure given a problem and heuristic, we can tell how well that heuristic conforms to the ideal policy on that problem. (Whether "promise" refers to the policy or to the measure based on this policy should be clear from the context.)

These points can be illustrated with a toy problem, in which promise can be worked out exactly for different consistency algorithms as well as for the entire problem. The problem consists of three variables, each with two or three values in its domain (see Figure 1).

For this problem, the search tree for simple backtracking (no filtering), using lexical variable ordering is:



To calculate promise, we consider the probability of choosing each viable value from a domain, when any value is equally likely to be chosen:



From this, we can calculate the overall promise for backtracking on this problem, by summing the path-products. The latter are:

$$\frac{1}{2}(\frac{1}{3})(0) = 0$$

$$\frac{1}{2}(\frac{1}{3})(\frac{1}{3}) = \frac{1}{18}$$

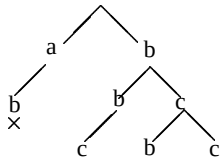
$$2\left(\frac{1}{2}(\frac{1}{3})(\frac{1}{3})\right) = \frac{2}{18}$$

And the resulting sum is:

$$0 + \frac{1}{18} + \frac{2}{18} = \frac{1}{6}$$

So the overall promise for this problem and this consistency algorithm is $\frac{1}{6}$. For backtracking, this is, in fact, the same as the solution density (cf. Section 5).

Now, we consider forward checking, using the SDF ordering. Here the search tree looks like:



Making the same calculations as before for this tree, we have the path-products,

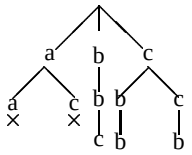
$$\frac{1}{2}(0) = 0, \quad \frac{1}{2}(\frac{1}{2})(1) = \frac{1}{4}, \quad \frac{1}{2}(\frac{1}{2})(1) = \frac{1}{4}$$

which gives the sum:

$$0 + \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

So the overall promise for this problem, given this variable ordering and the forward checking algorithm, is $\frac{1}{2}$.

In this case, if we use a different variable ordering, say 3-2-1 instead of 1-2-3, we, of course, get a different search tree:



Making the same calculations for this tree, we have path-products:

$$\frac{1}{3}(0) = 0, \quad \frac{1}{3}(1)(1) = \frac{1}{3}, \quad \frac{1}{3}(\frac{1}{2})(1) = \frac{1}{6}, \quad \frac{1}{3}(\frac{1}{2})(1) = \frac{1}{6}$$

and the sum:

$$0 + \frac{1}{3} + \frac{1}{6} + \frac{1}{6} = \frac{2}{3}$$

So the overall promise for this problem and this algorithm is $\frac{2}{3}$.

From this we can draw some important conclusions:

- Promise can vary depending on the variable ordering, as well as the consistency algorithm. (In the above three examples, three different values were obtained for the overall promise.)
- Promise is not in general equivalent to solution density; the equivalence holds only if there is no consistency maintenance.
- There does not appear to be a particular variable ordering that is guaranteed to give the best value for promise. (One might have expected SDF to do better than the alternative, and it may indeed on average.) Therefore, the degree of promise of different variable ordering heuristics will probably have to be decided empirically.

3.1 Probing for promise

Although the expected value of success gives an ideal measure of promise, it is not clear from the above whether such a measure can be obtained in general. Fortunately, it is possible to assess the overall promise of a problem under a given variable ordering with the following procedure, which we call *probing*. The basic idea is to choose, with appropriate randomization, variables and values until a dead end is encountered. At this point search stops and re-starts from the beginning with a new random seed. This process is repeated until a complete solution is found; the number of probes required to do this gives us a measure of promise, as shown below. Roughly, the greater the promise the fewer the number of probes required, on average, to obtain a complete solution.

This procedure avoids contamination by fail-firstness because we never try to recover from a deadend: the number of probes should be a reflection of promise alone. If used with a random value ordering heuristic over many runs, this technique also avoids effects of value selection on promise. We can use any consistency enforcement algorithm as part of the probing procedure. This allows us to assess this aspect of search for its effect on promise.

3.2 The probe procedure estimates promise

We can prove that the number of probes is a direct estimate of the promise for a problem, under a particular variable ordering. This is because the expected number of probes can be expressed by the following formula

$$\sum_{k=1}^{\infty} kp^{k-1}q$$

where q is the probability that the probe will succeed, and is therefore equal to the overall promise, and p is the probability that this probe

will fail. Using $1-p$ instead of q, we have.

$$\sum_{k=1}^{\infty} kp^{k-1}(1-p) \;=\; \sum_{k=1}^{\infty} kp^{k-1} - kp^{k}$$

This summation can be seen to involve a "telescoped" expression, so it can be simplified as follows. First, we consider the sum in its finite form,

$$[p^0 - p] \;+\; [2p - 2p^2] \;+\; [3p^2 - 3p^3] \;+\; \ldots +$$

$$[(n-1)p^{n-2} - (n-1)p^{n-1}] \;+\; [np^{n-1} - np^n]$$

where $n \rightarrow \infty$. This can be reduced to:

$$p^0 \;+\; p^1 \;+\; p^2 \;+\; \ldots +\; p^{n-1} \;-\; np^n$$

$$=\; (\sum_{k=0}^{n} p^k) \;-\; np^n$$

By a well-known equivalence, the summation within the parentheses is equal to

$$\frac{1-p^n}{1-p}$$

As $n \rightarrow \infty$, the simplified summation therefore goes to $\frac{1}{1-p}$, while the second expression (after the minus sign) goes to zero. The first statement is obvious. For the second, the following proof will suffice (and a variant can be used for the first statement as well).

Proof. The ratio of successive terms as n is incremented is:

$$\frac{(n+1)p^{n+1}}{np^n} \;=\; (\frac{n+1}{n})p$$

Since this ratio goes to p as $n \rightarrow \infty$, there must be some point at which it becomes less than 1, say, where $n = N$. Therefore, if we consider successive ratios from this point (all of which are now less than 1),

$$\frac{(N+1)p^{(N+1)}}{Np^N},\; \frac{(N+2)p^{(N+2)}}{(N+1)p^{(N+1)}},\; \ldots$$

the ratio of successive numerators in this sequence to the first denominator becomes progressively closer to 0. But then the original expression must $\rightarrow 0$. □

What this argument shows is that *the expected number of probes is equal to the reciprocal of the overall promise*. We can, therefore, use the probe technique to obtain a direct estimate of promise for any heuristic on any problem, as well as estimating the average promise of a heuristic for a class of problems.

For example, for forward checking using SDF on our toy problem, the summation is

$$\sum_{k=1}^{\infty} k(\frac{1}{2})^{k-1}\frac{1}{2} \;=\; 2$$

while for the reverse ordering, it is

$$\sum_{k=1}^{\infty} k(\frac{1}{3})^{k-1}\frac{2}{3} \;=\; 1.5$$

These are, of course, the reciprocals of the values of the overall promise calculated by hand above.

4 EMPIRICAL INVESTIGATIONS

In the last section, we have shown that on a toy problem different variable orderings result in different measures of promise and that *probing* allows us to estimate promise. In this section, we will use the probing procedure to investigate two hypotheses:

1. Different variable ordering heuristics from the literature exhibit different levels of promise.
2. Promise is inversely correlated with search effort. We do not expect promise to be perfectly correlated with search effort since fail-firstness surely also has an impact on search effort. However, we expect that for easier problems, promise will be strongly correlated with search cost. As problems become more difficult, we expect the effect of promise to decrease: fail-firstness should be more important to search effort.

4.1 Experimental details

We investigate the following set of heuristics and anti-heuristics:

- SDF: "smallest domain first". The variable chosen is the one with the smallest domain.
- max-static-degree: The variable chosen is the one with the maximum degree in the original constraint graph. This is a static heuristic since degree does not change during search.
- max-forward-degree: The variable chosen is the one with the maximum degree to non-assigned variables.
- brelaz [2]: The variable chosen is the one with the smallest domain. Ties are broken by choosing the variable with smallest domain and maximum forward degree.
- domdeg [1]: The variable chosen is the one that minimizes the ratio of domain size to forward degree (i.e. the number of adjacent uninstantiated variables).
- random: A random (unassigned) variable is chosen.
- LDF: "largest domain first". The variable with the largest domain is chosen.
- min-static-degree: The variable chosen is the one with smallest degree in the original constraint graph. This is a static heuristic.
- min-forward-degree: The variable chosen is the one with smallest degree to non-assigned variables.
- anti-brelaz: The variable with largest domain is chosen. Ties are broken by choosing the variable with largest domain which has minimum forward degree.
- anti-domdeg: The variable that maximizes the the ratio of domain size to forward degree is chosen.

Forward checking (FC) [6] is used with each variable ordering heuristic.

All test problems are randomly generated with 15 variables and 10 values per variable. The density is fixed at 0.7. Tightness varies from 0.30 to 0.39 in steps of 0.01. There are 100 problems in each subset corresponding to the different tightness values. The problems are not filtered and therefore (depending on the mean solubility of the set) contain a mix of soluble and insoluble problems.

To operationalize our probing technique for our experiments, we repeat the probing procedure 100 times for each problem and variable ordering heuristic with different seeds for the random number generator. The median reciprocal of the number of probes over the 100 runs is then considered our estimate of promise for that problem and variable ordering heuristic. Recall that for problems with no solutions, we define the promise to be 0. For a set of problems

and a variable ordering heuristic, we calculate the mean estimate of promise by finding the arithmetic mean of the promise estimate over each problem in the set for that variable ordering heuristic.

To estimate the search effort for a problem and a variable ordering heuristic, we follow a similar procedure as for estimating promise. The difference is that instead of probing for solutions, we simply use chronological backtracking. Our measure of search effort is the number of consistency checks required to find a solution. Again, for a given problem and variable ordering heuristic, we run this process 100 times with differing random seeds and define the search cost to be the median number of constraint checks over the 100 runs. For a set of problems the mean search cost is the arithmetic mean of the search cost for each problem.

4.2 Results

Figure 2 presents the mean promise estimates for each variable ordering heuristic. A log-scale is used on the y-axis to make the rankings of the heuristics easier to see. Figure 3 presents the search cost for each variable ordering heuristic.

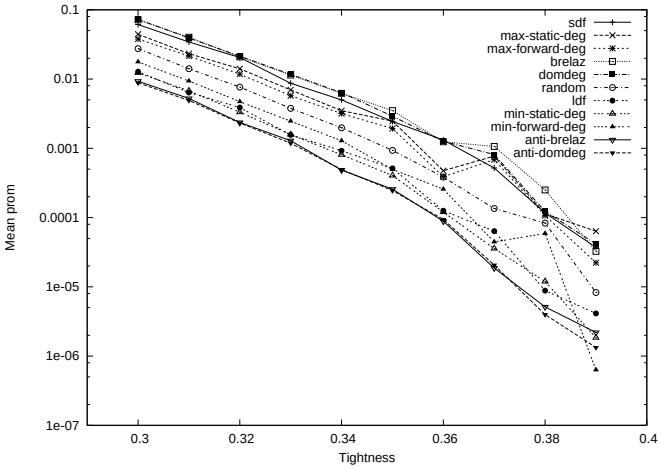


Figure 2. The mean promise estimates for each set of problems.

Figure 2 clearly shows that different variable ordering heuristics exhibit different levels of promise. As the tightness increases, there are fewer soluble problems and the level of promise correspondingly decreases. Even when we remove the insoluble problems, however, the promise decreases with increasing tightness as fewer solutions result in a lower promise for a given variable ordering heuristic.

Comparing Figures 2 and 3, we can see that the most successful heuristics (i.e., those with lowest search cost: domdeg, brelaz, and SDF) also exhibit the highest levels of promise. Furthermore, the rankings seem relatively consistent: with some exceptions, the i^{th} best heuristic exhibits the i^{th} highest level of promise.

Figure 4 presents a measure of the correlation between promise and search cost. The plot labeled "all" examines 1100 points for each problem set composed of the search cost and promise values for each of the 100 test problems and each of the 11 variable ordering heuristics. The plot shows that for low values of tightness, the variation in promise accounts for 70-80% of the variation in search cost. As the problems become tighter the R^2 value drops to close to 0. This is consistent with our expectations.

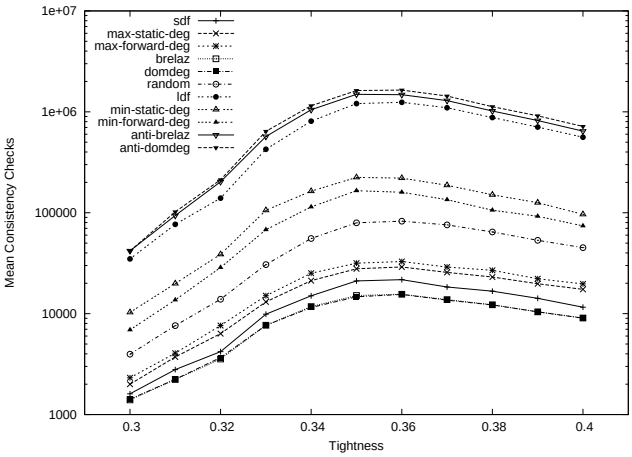


Figure 3. The mean search cost for each set of problems.

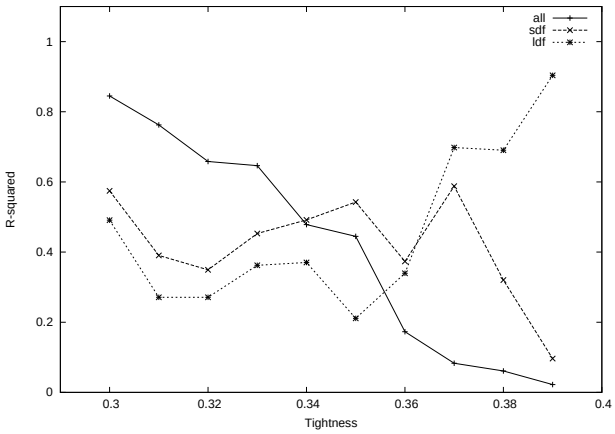


Figure 4. The R^2 values for the correlations between promise and the reciprocal of search cost for each set of problems and for selected variable ordering heuristics.

Figure 4 also presents the R^2 values for SDF and its anti-heuristic, LDF. This plot is similar to that seen with the other heuristic/anti-heuristic pairs based on domain size (i.e., domdeg and brelaz): at low values of tightness the R^2 values are somewhat noisy and around 0.4. For the tighter problem sets, the anti-heuristic, LDF, starts to have a much stronger correlation with search cost while the correlation of the heuristic, SDF, declines. This is somewhat surprising as we would expect the promise of a heuristic to be quite poorly correlated with search cost when the problems are insoluble. However, a closer look at the data shows that for problems with no solutions, the anti-heuristics have quite a small variance in search cost. The anti-heuristics do poorly, of course, but each anti-heuristic appears to do uniformly poorly across the insoluble problems at a tightness value. Given that there are very few soluble problems at high tightness values (i.e., one for set 0.39) this leads to a strong correlation between promise which is, by definition, 0, and search cost. Another surprising result is that for the problem sets with low tightness, the R^2 values are much lower than the "all" plot. In other words, the relationship between promise and search cost is weaker within any single heuristic, but over all heuristics the trend of higher promise corresponding to lower search cost for loose problems is clear.

The heuristics and anti-heuristics based only on variable degree have a similar behaviour as the domain-based heuristics for low values of tightness. As the tightness increases, however, the R^2 values of both the heuristics and anti-heuristics drop as expected. Lending support to our above explanation of the domain-based anti-heuristics, the degree-based anti-heuristics have a larger variance on insoluble problems than their domain-based counterparts.

5 DISCUSSION AND OBSERVATIONS

We have shown that promise has a high (inverse) correlation with search effort for problems with many solutions but a low correlation when the number of solutions decreases. We have also shown that the number of probes to find a solution correlates with the effort for complete search to find a solution. But why should heuristics such as SDF, domdeg, and brelaz show greater degrees of promise than other variable orderings? These heuristics give preference to variables with small domain sizes. If we assume that each value in the domain of a variable has equal probability of occurring in a solution, when domain sizes are small the probability of any value in that domain being in a solution is relatively high. That is, the values are more promising. Therefore we should expect that heuristics that prefer variables with small domains will be promising, and this is just what we have seen. This explanation may also account for the differences between heuristics and anti-heuristics that are based on degree alone, although this has not yet been tested.

These results should not come as a surprise. What we have demonstrated is that the extent to which a variable ordering affects the likelihood of finding a solution has an inverse relationship to the overall search effort. Qualitatively, this is only to be expected. But heretofore it has not been clear that differing degrees of promise actually play a (non-negligible) role in performance differences based on variable ordering. In fact, prior to this work no means of analysis has been available to address the issue. It is certainly conceivable that the ability to escape dead ends is so important that search cost depends entirely on fail-firstness and the promise of a heuristic is irrelevant or, at best, of marginal importance. Indeed, the focus on the fail-first principle in the literature and the fact that promise has, to our knowledge, never been explicitly investigated, might lead one to this expectation.

In our toy example we demonstrated that when using forward checking, different variable orderings have different levels of promise. When calculating the promise for the entire tree we are only interested in paths to solutions (all other paths are zero valued). To increase promise we want to reduce the degree of nodes in that tree, and we can do this by increasing the level of consistency during search, i.e. increasing levels of consistency will be increasingly promising. If our search process does not perform any domain filtering, but checks backwards (i.e. the algorithm is BT), then all variable orderings will have the same promise. Consider a variable X. This will correspond to an interior node in the search tree with out degree equal to the domain size of that variable. Each value will then be equally promising. In any rearrangement of the tree X will have the same out degree, and each value will have the same promise. Therefore, all heuristics will be equally promising. This reasoning has been confirmed empirically, with the problems used in the experiments above.

The likelihood that a value in the domain of a variable is in a solution will increase as we increase the level of consistency. At the extreme, with n-consistency, all domain values are in solutions, and again all heuristics will be equally promising. However, between the extremes of BT and n-consistency we should expect that as consistency levels increase the difference in promise between heuristics will decrease. Therefore, promise should actually be defined on the triple $\langle problem, heuristic, consistency\ level \rangle$.

6 FUTURE WORK AND CONCLUSION

Symmetrical to promise will be fail-firstness, i.e. the ability of a heuristic to detect a bad decision. Since heuristics are fallible, when problems are soluble both promise and fail-firstness will come into play, and when problems are insoluble promise will be irrelevant. We conjecture that variable ordering heuristics will exhibit differing levels of fail-firstness as well as promise. While formalizations of the fail-first principle have been made [6, 7], no one, to our knowledge, has developed a method for measuring fail-firstness that is independent of promise. We plan to apply the same methodology used with promise to fail-firstness and then to evaluate the extent to which search cost is correlated with the combination of promise and fail-firstness.

Why did Smith and Grant's aggressive attempts to fail earlier [8] result in poor search performance? It could be that as they increased fail-firstness, they reduced the heuristic's promise. We plan to empirically test this hypothesis. Furthermore, it would be interesting to see if we could design stronger heuristics by investigating the trade-off between promise and fail-firstness and, ideally, developing a heuristic with higher promise and higher fail-firstness.

We have presented what we believe to be an explanation of what makes DVO heuristics perform well, i.e. promise and fail-firstness. Promise guides search to a solution when one exists and fail-firstness delivers short proofs of insolubility. Variable ordering heuristics, such as SDF, brelaz and domdeg exhibit promise as well as fail-firstness. Why is this interesting? Up till now we have had conflicting guidance on what makes a good heuristic, making it difficult to design new heuristics. The present approach offers a way out of this impasse by developing a more thorough analysis of heuristics starting from first principles in the form of optimal policies.

ACKNOWLEDGEMENTS

This work was supported by Science Foundation Ireland under Grant 00/PI.1/C075.

REFERENCES

[1] C. Bessiere and J-C. Regin, 'MAC and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems', *Principles and Practice of Constraint Programming-CP'96*, 61–75, (1996).
[2] D. Brélaz, 'New methods to color the vertices of a graph', *Communications of the ACM*, **22**(4), 251–256, (1979).
[3] J. Cheverton, A. Kacelnik, and J. R. Krebs, 'Optimal foraging: constraints and currencies', in *Experimental Behavioral Ecology and Sociobiology*, eds., B. Hölldobler and M. Lindauer, 109–126, Sinauer, Sunderland, MA, (1985).
[4] P. A. Geelen, 'Dual viewpoint heuristics for binary constraint satisfaction problems', in *Proc. 10th European Conf. Artif. Intell.*, pp. 31–35, (1992).
[5] I.P. Gent, E. MacIntyre, P. Prosser, B.M. Smith, and T. Walsh, 'An empirical study of dynamic variable ordering heuristics for the constraint satisfaction problem', *Principles and Practice of Constraint Programming-CP'96*, 179–193, (1996).
[6] R. M. Haralick and G. L. Elliot, 'Increasing tree search efficiency for constraint satisfaction problems', *Artif. Intell.*, **14**, 263–314, (1980).
[7] B. Nudel, 'Consistent-labeling problems and their algorithms: Expected-complexities and theory-based heuristics', *Artif. Intell.*, **21**, 263–313, (1983).
[8] B. M. Smith and S. A. Grant, 'Trying harder to fail first', in *Proc. 13th European Conf. Artif. Intell.*, pp. 249–253, (1998).
[9] E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, 1993.