Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference

A Normative-Prescriptive-Descriptive Approach to Analyzing CSP Heuristics

Richard J. Wallace
Insight Centre for Data Analytics, Department of Computer Science
Western Gateway Building, University College Cork, Ireland
richard.wallace@insight-centre.org

Abstract

This paper presents a general framework for analyzing heuristics for constraint solving, including backtracking and arc consistency algorithms. It will emphasize heuristics for variable selection during search, since this is where major differences are found. In earlier work two basic approaches to this problem were developed. The first was a general theoretical framework for different types of heuristics, which characterized ideal performance so that the actual performance of heuristics could be compared to this standard. The second involved the discovery that, while there are a large number of features that can be used for heuristic decisions in variable ordering, differences in effectiveness boil down to only two basic "heuristic actions". The present paper applies basic ideas from decision analysis to characterize these two approaches to better understand their status and interrelations. It shows that the first is essentially a normative decision analysis, and that models of this sort imply general prescriptive principles (notably the Fail-First Principle). The second is concerned with descriptive models of actual performance.

Introduction

Although heuristic decisions play a critical role in search algorithms for constraint satisfaction problems (CSPs), their analysis has lagged behind other aspects such as the analysis of local consistencies. In fact, the evaluation of CSP heuristics, especially for variable ordering, is an area of constraint programming that has remained strangely resistant to the kind of formal analysis that has come to characterise the field overall.

The present paper argues that the analysis of heuristics is a much different problem than the usual analysis of algorithms. Therefore, it requires different methods. This paper will outline a general framework for heuristic analysis, inspired by the scheme that has grown up around the analysis of decision making in the decision sciences. This will serve to clarify the various research strands that exist in the literature, whose relations have heretofore remained fairly obscure. And with a coherent framework in hand, one can see more clearly where the gaps are that remain to be filled in.

In the present work the focus is on variable ordering heuristics for complete algorithms, both because of their signal importance in CSP search and because their analysis is especially interesting. However, I will touch on heuristics for value ordering as well as queue ordering for certain arc consistency algorithms. The framework presented here can also be extended to branching rules for Boolean satisfiability (SAT) algorithms, although they will not be discussed here.

Most of the work on variable ordering heuristics has concerned itself with finding new and better heuristics. In this work, little or no thought has been given to understanding and analysing why some heuristics lead to greater efficiency, or even to how an analysis might be carried out. However, over the years there have been occasional contributions that are pertinent to untangling this problem.

Perhaps the first example was the introduction of the Fail-First Principle (Haralick and Elliott 1980). Interestingly, until recently the status of this principle was not clear; in some cases, it has been confused with the minimum domain heuristic, which has been called the fail-first heuristic. As will be seen, there is a sense in which minimum domain can be construed as *a* fail-first heuristic, but that does not make it equivalent to the Fail-First Principle.

The next relevant contribution occurred in connection with "branching heuristics" for SAT problems. (Branching heuristics in fact refer to both variable and value ordering; here, we emphasize the former.) To explain a rule based on a formula that favoured literals appearing in many short clauses, it was suggested that it worked because it created subproblems that were more likely to be satisfiable (Jeroslow and Wang 1990). Later this rationale was questioned (since the rule worked even for unsatisfiable problems); the latter authors suggested that branching rules improve search efficiency by creating "simpler" subproblems. By simplification, they meant the degree that the size of the formula was reduced (Hooker and Vinay 1995).

Toward the end of the 1990s, Smith and Grant carried out an ingenious analysis of 'fail-firstness' (Smith and Grant 1998). After deriving a formula that predicted immediate failure after variable selection, they devised a set of heuristics based on successive approximations to the formula, beginning with minimum domain. Since the relative performance of the heuristics did not match the degree of approximation, they concluded that the Fail-First Principle was not a sufficient explanation for differences in performance due to heuristic choice.

During the last decade, Beck et al. introduced what they

called the Policy Framework for analyzing CSP variable ordering (Beck, Prosser, and Wallace 2003; 2005). This was meant to extend earlier ideas about fail-firstness in order to provide a fuller account. A few years after that, Wallace presented work that analysed heuristic performance using statistical techniques for discovering underlying "factors" that accounted for differences in heuristic performance (Wallace 2005). This work led to the notion of "heuristic actions"; roughly speaking, these are effects on features of search when a given heuristic is used that can have marked effects on algorithm performance.

In this work there are still many loose ends to tie up. However, in the last decade there has been little or no progress at the level of conceptualization. Instead, the field has continued using empirical, seat-of-the-pants approaches, concocting new strategies, such as search with heuristic profiles or various statistical approaches to heuristic decisions (e.g. (Phillips et al. 2015)). Part of the problem may be that the various *fundamental* contributions to the analysis of CSP heuristics have not been put together into one coherent framework. So it has been difficult for workers in the field to get a handle on this material or to see how it might be extended. This is the motivation for the present paper.

Here, I argue that the status and character of these various contributions can be greatly clarified by placing them within a decision theoretic framework. This will show how they are related to each other. It also shows that we now have the skeleton of an account of heuristics in this domain that is in some sense complete. In a real sense, this is the piece of the puzzle that allows us to arrange the other pieces into a coherent whole.

Background Concepts for CSPs

A constraint satisfaction problem (CSP) is defined as a tuple (V, D, C) where: $V = \{V_1, ..., V_n\}$ is a set of variables which must be assigned values; $D = \{D_1, ..., D_n\}$ is a set of domains, where D_i is the set of possible values which may be assigned to the variable V_i; and $C = \{C_1, ..., C_m\}$ is the set of constraints that indicate which combinations of values can go together in a solution. A solution to a problem is an assignment of a value to every variable such that no constraint is violated.

Complete algorithms in this area are all extensions of depth-first backtrack search. This procedure involves decisions at various points that are not specified algorithmically, hence the opportunity for devising heuristics in order to make decisions that speed up search. In addition, the typical interleaving of search and local consistency processing involves other non-deterministic decisions. In this paper, we are mostly concerned with heuristics for selecting the next variable to assign a value to. This is because the order in which variables are selected for instantiation during search can have a marked effect on overall performance.

Heuristics are based on features of the situation that serve to distinguish choices. Examples of such "rules" for variable ordering include selecting the variable with the smallest domain first, or the variable with the maximum degree or one for which the ratio of domain size to degree is minimized. Specific rules for value selection include the minimum number of conflicts, or the maximum minimum domain size after some form of propagation, etc. The rationale usually given for variable ordering heuristics is based on the Fail-First Principle (discussed below), while that for value ordering is related to the total number of values still allowed after a particular value is chosen.

Relevant Concepts from Decision Analysis

In the study of decision making it is usual to divide the field of endeavour into three parts, called *normative*, *prescriptive*, and *descriptive* decision analysis. Normative analysis is concerned with characterising rational decision making according to first principles. Descriptive decision analysis is concerned with decision making as an empirical phenomenon. It tries to characterize how decision makers actually go about making decisions, either under laboratory conditions or in the field. Prescriptive decision analysis falls in between the other two. It is concerned with the application of formal or semi-formal procedures, based in part on assumptions about rationality, in order to help the decision maker make real-world decisions in line with rational principles.

As the reader undoubtedly knows, the central concept in decision analysis is that of preference. Usually, this is represented by a preference relation over a set of alternatives. Secondarily, there may be a utility function defined over the same set and a representation theorem relating the function to the preference relation. As far as I can discern, none of this apparatus is especially useful in the present domain. This is simply because preferences do not have to be defined apart from the basic efficiency measures, which can be assessed directly. So all that is needed for our purposes is the basic tripartite distinction described above. This curious state of affairs will be clarified as I describe the different lines of work alluded to in the Introduction.

A Normative Framework for Variable Ordering Heuristics

For search problems, there is an overall goal of minimizing search effort in terms of the number of decisions that must be made. This, therefore, provides the basis for any sensible framework for evaluating heuristics. For variable ordering heuristics, the most straightforward global measure of search effort is the number of nodes in the search tree. A "search node" is a partial instantiation of the variables. Thus, every time an assignment is extended by assigning a value to another variable (variable k+1), and every time the current variable (k) is given a different assignment, we consider that an additional search node has been generated.

Other measures of search effort have been suggested, such as number of backtracks or number of search nodes associated with a domain wipeout. But these measures are either not monotonic in the number of search nodes or simply give a subset of nodes without explaining why this is superior to counting all of them.

Moreover, all these methods of counting are simple empirical measures of search effort. Next I will outline an alternative approach, based on an abstract model of perfect search. This allows us to evaluate performance quality in a

way that is much more general. More importantly, on this basis we can develop a more articulated assessment of search effort grounded on a rational model.

A new framework based on ideal policies

Some years ago (Beck, Prosser, and Wallace 2003) introduced what they called the *Policy Framework* as a means of characterising performance quality when variables are chosen using a given heuristic. This framework has two basic elements, called policies and heuristics. Both are concerned with selecting the next variable to instantiate during search. Heuristics are defined as before. Policies, on the other hand, define decisions in terms of goals or end-results. Moreover, policies are to be construed as characterizing ideal (or if you like, optimal) decisions. They can be thought of as carried out under conditions of perfect information regarding which choice should be made *at a given point in search*.

From the perspective of decision analysis, the Policy Framework is *a normative model of variable selection*, which describes how decisions should be made under ideal conditions. As with all normative models of decision making, the purpose of this approach is to characterize the quality of actual decision rules in terms of a reasonable ideal. This is analogous to the situation in ordinary Decision Theory where one makes decisions in accordance with a well-defined utility function. Here, the model describes an ideal case in which one has complete information about the preferability of each possible choice.

Now, what are the actual policies? In the present context, where the decision is what variable to choose next for assignment, two policies can be distinguished, which depend on the state of search. When search is in a state that has solutions in its subtree, effort will be minimized by making decisions that remain on a path to a solution. As this involves making decisions that are most promising in that regard, this is called the *promise policy*. However, when search is in a subtree that does not contain solutions, this policy obviously cannot hold. In this case, to minimize search effort, choices should be made so as to fail as quickly as possible so that search can return to a path that leads to a solution. This is called the *fail-first policy*.

Notice that the two policies are based on a partition of search nodes into those that have solutions in their subtree ("good" nodes) and those which do not ("bad" nodes). If the partition that a node is in is known, the policy which leads to minimal search effort is given. Achieving that goal in practice usually involves heuristics because, (i) typically since we do not know whether the current node is good or bad, we do not know which policy to adhere to, (ii) even knowing a policy, we do not know *how* to adhere to it.

The Fail-First principle

Harralick and Eliot were the first to knowingly apply the Fail-First Principle to CSP search (Haralick and Elliott 1980). Their original statement of the principle was (p. 302):

> The fail first principle states that we should first try those tests in the given set of tests that are most likely to fail since if they do fail we do not have to do the remainder of tests in the set.

In the context of CSP search, a "test" is a choice of the next variable to make assignments to.

First of all, it should be noted that the Fail-First Principle is *not* the same as the fail-first policy. From the perspective of decision analysis, the Fail-First Principle is a general prescriptive rule rather than a search heuristic, although as noted earlier this has not always been recognized. When Harralick and Eliot introduced their prescriptive rule, there was no normative theory to serve as a framework. But given the Policy Framework, we can recast this principle in terms of the policies we have described, thus showing more clearly what it is meant to convey.

Put in these terms, the principle says that in choosing the next variable to instantiate, one should always act as if the fail-first *policy* is in force. Under this assumption, a good heuristic is one that conforms to this policy. In this form, the Fail-First Principle can be viewed as a kind of *ideal* metaheuristic that specifies which policy to take into account during search. It is ideal because one does not actually know which policy is in force at a given point during search. Nonetheless, it can serve as a general guide in heuristic design. This was demonstrated by Harralick and Eliot when they showed that min domain size has properties that conform to general expectations under the Fail-First Principle.

The present analysis is the first to place this well-known principle within a general framework and thus to give it a coherent characterization. This is further evidence that the Policy Framework is a sensible one as well as showing the usefulness of formulating a normative model.

Policies and performance measures

The policy framework is intended to provide a more coherent analysis of the quality of heuristic performance than can be provided by measures of overall effort alone. But for this purpose, measures are needed to assess the degree to which a heuristic adheres to each policy.

A measure of adherence of a variable ordering heuristic to the promise policy must be based on the mean likelihood of choosing a value that will lead to a solution across all paths in the (all-solutions) search tree (Beck, Prosser, and Wallace 2003). Although we initially used Monte Carlo methods to estimate such probabilities, a better measure can be obtained by carrying out an exhaustive search while collecting sums of products that are returned at successively higher levels of the search tree. Summing is done across the values at a given level of search, and products are taken along search paths.

For the fail-first policy, an adequate measure must be based on the average size of the subtree associated with an assignment that *does not* lead to a solution, i.e. the average size of an insoluble subtree rooted at the first bad assignment. This is called a "mistake-tree" to distinguish it from insoluble trees in the ordinary sense (Beck, Prosser, and Wallace 2005). By specifying the root as the first 'bad' assignment, we produce an intensity measure, and we are able to compare heuristics across soluble and insoluble problems with respect to the intensity of fail-firstness.

Other candidate measures of fail-firstness such as average depth of failure and number of failures are affected by promise as well as fail-firstness and for this reason are not

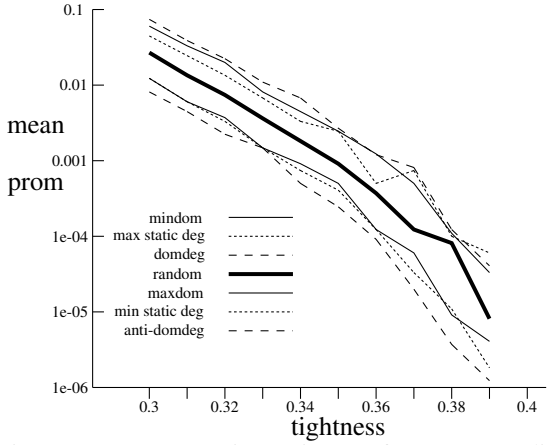

Figure 1: Mean promise estimates for corresponding variable ordering heuristics and anti-heuristics.

adequate. Fail-length, which is the difference in depth between the initial mistake and an actual failure, avoids this problem; since it is an average value, it is also a true intensity measure. However, tests have shown that it is essential to take the branching factor into account in measuring fail-firstness as well as rapidity of failure (Beck, Prosser, and Wallace 2005).

It is not yet clear how best to combine such data to make a global measure, as we can with the promise measure. To date, it has been found sufficient to consider mean mistake-tree size for single levels or averaged across the first k levels.

Insights obtained with the Policy Framework

These tools can be used to demonstrate basic properties of heuristics as well as elucidating specific puzzles. An example of the former is the demonstration made early in our investigations that in general good heuristics not only have superior fail-firstness but also superior promise. This is shown in Figure 1, based on (Beck, Prosser, and Wallace 2003). This means that *a complete account of heuristic performance must pay attention to both policies*; the fail-first policy by itself does not give a complete account of why one heuristic outperforms another. Note also that this in itself does not contradict the prescriptive Fail-First Principle, although it suggests that the Principle may have limitations in practice.

Occasionally, one finds that a heuristic based on a complex of problem features shows good fail-firstness and at the same time poor promise. This occurs in the fail-first series of (Smith and Grant 1998). These authors derived a series of heuristics (called min-domain, FF2, FF3 and FF4) that gave successively better approximations to a formula for the likelihood of failure. Nonetheless, when these were used with the forward-checking algorithm, performance did not match the degree of approximation. With the policy measures it can be shown that fail-firstness does improve successively as expected, but FF3 and FF4 show poor adherence to the promise policy in comparison with FF1 and 2 (Table 1, from (Wallace 2006)). These results also show that, contrary to initial expectations, adherence to the promise policy is important

	FC		MAC	
heur	promise	ff	promise	ff
dom	.0007475	206	.006807	30.9
ff2	.0006783	115	.007337	19.0
ff3	.0000028	87	.007221	16.5
ff4	.0000002	47	.007494	12.7

Note. <30,8,0.31,0.34> problems. Means for 100 problems. ff measure is mean mistake-tree size.

Table 1: FF Series: Policy Adherence with FC and MAC

even when problems are in a critical complexity region.

Descriptive Analyses of Heuristic Performance

The normative-prescriptive framework just described does *not* give direct insight into why heuristics actually work. Given the scheme proposed here, we can see why this is so; it is because this question must be answered at the descriptive level of decision analysis. But to answer this question properly, we must consider more carefully what the actual problem is.

Heuristic features versus heuristic effects

Almost all work in the field of heuristic studies has focused on finding metrics that can be used to make heuristic decisions. An important early discovery was that features can be either static or dynamic, the latter referring to features that change during the course of search. A standard static feature is the degree of a variable. Two classical dynamic features are the current size of a variable's domain and the "forward degree" of a variable, that is, the number of adjacent variables not yet assigned values. One of the most interesting early empirical discoveries is that great improvements in efficiency can be obtained when metrics are based on dynamic features such as these. A major recent advance in this department was the introduction about ten years ago of adaptive heuristics such as impact or weighted degree (Boussemart et al. 2004; Refalo 2004). These heuristics use information gathered during search in order to make finer discriminations among variables than is possible with either static or dynamic features.

However, none of this work serves to explain *why* heuristics are effective. To answer this question properly, we must first deal with a more general question, which is: how many ways are there in which variable ordering heuristics can improve search? Given that several dozen variable ordering heuristics have been proposed over the past 50 years, one can pose a question that is roughly equivalent: to what degree are these heuristics actually doing different things in order to improve search? Or are they doing the same thing with varying degrees of efficiency?

Factor analytic studies of heuristic action

Answers to these questions can be obtained using the well-known statistical technique of factor analysis (FA). Most people in AI know this method mainly through principle components analysis. The difference between the two is that

with FA one is trying to find a particular model with m factors that explains an appreciable amount of the variance in n measures, where $m \ll n$.

Patterns of performance across a set of (homogeneous) random binary CSPs having the same basic parameter values (i.e. same number of variables, domain size, etc.) for the same algorithm, but with twelve different heuristics, were subjected to factor analysis. By combining this technique with various experimental controls, it was possible to show that over 90% of the variance in the results could be explained with only two factors (Wallace 2005; 2008). All heuristics tested showed higher correlations with one or the other of the two factors. Thus, heuristics could be classified according to the factor they 'favoured'. It is critical to note, however, that there were usually definite positive loadings on both factors (typically, .45 versus .85 for a given heuristic).

Now, since all that factor analysis does is extract patterns of variation, further analysis (both conceptual and experimental) is necessary to determine the cause of this variation and whether it is of *scientific* significance. In the present case, there is compelling evidence that these two factors reflect two distinct *heuristic actions*. The distinctness is also indicated by the fact that factor analysis in its usual form extracts uncorrelated factors.

The most compelling evidence for this comes from experiments in which heuristics were based on additive combinations of rankings derived from two or more of the basic heuristics used in the original experiments (Wallace 2006). It was shown that combinations of heuristics that loaded most heavily on the *different* factors served to enhance search, so that performance was better than the best heuristic used singly. This effect was called "heuristic synergy". If heuristics favoured the same factor, results for the combination were in between those for the heuristics used singly. Moreover, synergistic effects were as great for appropriate combinations of two heuristics as for combinations of three or more, which supports the hypothesis that there are only two basic kinds of heuristic action for these problems (and possibly only two *fundamental* heuristic actions in general).

Table 2. Development of Promise with Successive Correct Assignments for Different Heuristics

		level				
heur	mean nodes	0	1	2	3	4
d/dg	103	.083	.167	.500	1.000	1.000
fd	69	.048	.333	1.000	1.000	1.000

A <30,8,0.31,0.37> problem with two heuristics.
"level" is assignments before promise calculation.

Further evidence comes from the fact that the two heuristic actions have different performance signatures (Wallace 2008). That is, measures such as average depth of failure or average branching factor show striking differences between the two groups. In addition, although the usual fail-first measure does not distinguish between the two actions, the promise measure can be used to discriminate between them. If one measures promise at successive levels of search, one finds that the rate of increase in this measure is much greater for the forward-degree type of heuristic, and again there is no overlap in the averages. Thus, with 100 30-variable problems, the mean search level at which the promise measure first equalled 1.0 varied between 4.5 and 9.5 for the min-domain type of heuristics and 3.5 and 4.1 for forward-degree heuristics. An example for one problem in this set is shown in Table 2 from (Wallace 2008).

It is possible to associate one of these two heuristic actions with the simplification idea of (Hooker and Vinay 1995); hence, it is referred to as the "simplification factor". The other factor is referred to as the "contention factor" (Wallace 2008). Another way of describing these actions is to say that the latter focuses on immediate failure (i.e. failure of the current variable), while the former focuses on future failure (Wallace 2005).

Note that the proposal is not that there are two classes of heuristics, although an important question is why heuristics tend to fall into two classes with respect to basic heuristic actions. In fact, a heuristic based on more than one problem feature may be more highly associated with one or the other action depending on the kind of problem. Thus, two heuristics in the aforementioned FF series of (Smith and Grant 1998), FF2 and FF3, are contention heuristics when used on random CSPs and simplification heuristics when used on random k-colouring problems (Wallace 2008).

Figure 2 shows a comparison between a 'simplification heuristic' (max forward degree, fd) and a 'contention heuristic' (the fail-first heuristic ff2), which have roughly similar efficiency in a given sample of homogeneous random CSPs. Here, two samples were culled from a larger set of problems; these were the problems in which one type of heuristic or the other was most strongly favoured. Each heuristic has a characteristic performance signature which is found whether or not it performs efficiently. For the simplification heuristic most of the work is done at lower levels of the search tree, while for the contention heuristic most work occurs at deeper levels and is more spread out. When problems are amenable to one heuristic action the curve of effort is greatly reduced, but the shape of the curve remains the same.

All this should not be taken to mean that different heuristics are not useful for different problem classes. This is because with problems with heterogeneous features different parts of the problem may be differentially amenable as well. In particular, heuristics that use different metrics may select different parts of the problem at the beginning of search. Then, if that part of the problem is amenable to the kind of heuristic action produced by that heuristic, then these heuristics will outperform other heuristics that either do not choose this part of the problem first or do not effect an appropriate action. Evidence for this has been obtained using heterogeneous random problems (Wallace 2008).

Other Kinds of Heuristics

The same tripartite approach presented here may also be useful in the analysis of, (i) value ordering heuristics, (ii) heuristics for selecting the next queue element in certain arc consistency algorithms such as AC-3. In the latter case, what was in fact a prescriptive rule was suggested by Waltz when

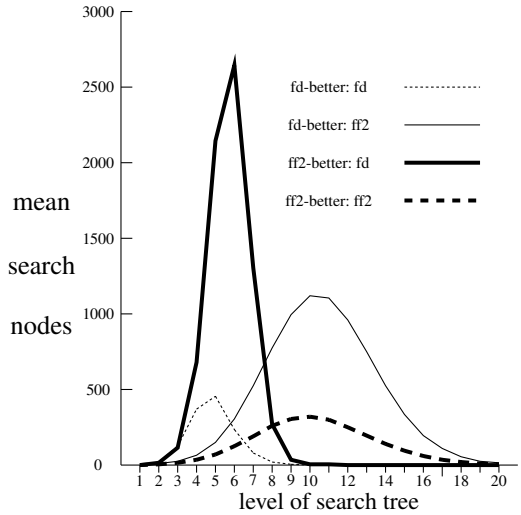

Figure 2: Search nodes as a function of depth of search for two heuristics and two problem sets. For each heuristic, the higher curve is for performance on the set of problems that it did poorly on. Hence, each *problem* set is associated with a higher curve and a lower curve of the alternative type.

he introduced the idea of arc consistency (Waltz 1975). The rule was simply to delete values as quickly as possible in order to process the problem more efficiently. This was renamed the ASAP Principle by (Wallace and Freuder 1992), and although there it was distinguished from AC heuristics by calling it a "performance principle", its status as an ideal meta-heuristic was not yet recognized.

Concluding Comments

With this decision theoretic framework, we can now organize previous advances in the scientific analysis of CSP heuristics in a coherent fashion. Of course, we are still far from the ultimate goal of heuristic design from first principles. Nonetheless, this goal can only be achieved after developing a coherent framework. The claim made in this paper is that the basic elements of this framework are now in place.

However, there are still many unanswered questions and major gaps to be filled. Perhaps the most important question is the nature of these heuristic actions. As yet, our understanding is largely intuitive. An answer to this fundamental question should also answer the following:

- What is the basis for the characteristic signature of each of the two types of heuristic action?
- Why is balancing heuristic actions so effective and why do adaptive heuristics facilitate this balancing?

There are also questions about the relation between the normative model and the number and nature of the heuristic actions that are still unresolved. As already indicated, the relation between these levels of analysis is not straightforward. It may be that significant relations exist with respect to dynamic changes in promise and fail-firstness as search continues, but this is a line of inquiry that has not yet been explored.

Finally, I think it is interesting in itself that the classical tripartite decision theoretic framework has proven to be useful in this context. It raises the question of whether a similar approach would be useful in other cases of algorithmic nondeterminism.

References

Beck, J. C.; Prosser, P.; and Wallace, R. J. 2003. Toward understanding variable ordering heuristics for constraint satisfaction problems. In *Proc. Fourteenth Irish Conference on AI & Cognitive Science-AICS'03*, 11–16.

Beck, J. C.; Prosser, P.; and Wallace, R. J. 2005. Trying again to fail-first. In Faltings, B.; Petcu, A.; Fages, F.; and Rossi, F., eds., *Recent Advances in Constraints-CSCLP 2004. LNAI No. 3419*, 41–55. Springer.

Boussemart, F.; Hemery, F.; Lecoutre, C.; and Sais, L. 2004. Boosting systematic search by weighting constraints. In *Proc. Sixteenth European Conference on Artificial Intelligence-ECAI'04*, 146–150. IOS.

Haralick, R. M., and Elliott, G. L. 1980. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence* 14:263–314.

Hooker, J. N., and Vinay, V. 1995. Branching rules for satisfiability. *Journal of Automated Reasoning* 15:359–383.

Jeroslow, R., and Wang, J. 1990. Solving propositional satisfiability problems. *Annals of Mathematics and AI* 1:167–187.

Phillips, M.; Narayanan, V.; Aine, S.; and Likhachev, M. 2015. Efficient search with an ensemble of heuristics. In *Proc. Twenty-Fourth International Joint Conference on Artificial Intelligence – IJCAI'15*.

Refalo, P. 2004. Impact-based search strategies for constraint programming. In *Principles and Practice of Constraint Programming-CP 2004. LNCS No. 3258*, 557–571. Springer.

Smith, B. M., and Grant, S. A. 1998. Trying harder to fail first. In *Proc. Thirteenth European Conference on Artificial Intelligence-ECAI'98*, 249–253. Wiley.

Wallace, R. J., and Freuder, E. C. 1992. Ordering heuristics for arc consistency algorithms. In *Proc. Ninth Canadian Conference on Artificial Intelligence - CAI'92*, 163–169. Morgan Kaufmann.

Wallace, R. J. 2005. Factor analytic studies of CSP heuristics. In *Principles and Practice of Constraint Programming-CP'05. LNCS No. 3709*, 712–726. Springer.

Wallace, R. J. 2006. Analysis of heuristic synergies. In Hnich, B.; Carlsson, M.; Fages, F.; and Rossi, F., eds., *Recent Advances in Constraints-CSCLP 2005. LNAI. No. 3978*, 73–87. Springer.

Wallace, R. J. 2008. Determining the principles underlying performance variation in CSP heuristics. *International Journal on Artificial Intelligence Tools* 17(5):857–880.

Waltz, D. L. 1975. Understanding line drawings of scenes with shadows. In Winston, P. H., ed., *The Psychology of Computer Vision*. McGraw-Hill. 19–91.