Neural Network Based Analysis of Amenability of CSPs to Different Heuristic Actions

Richard J. Wallace

School of Computer Science and IT, University College Cork, Ireland

Abstract

Earlier work has shown that variable ordering heuristics for CSPs improve search by enhancing two fundamentally different actions, labeled simplification and contention. The present work deals with the corresponding problem of why different problems in a supposedly homogeneous set respond differentially to these two heuristic actions. To this end, deep neural networks were tested to determine whether they could learn to discriminate among problems on this basis given a particular set of feature measurements. Following the success of these tests, explainable AI methods were used to gain some insight into the basis for this capacity. The most important features involved the structure of the contraint graph, in particular the degree distribution across variables. However, further tests showed that a number of other features had to be incorporated for highly accurate classification.

Keywords

constraint satisfaction problems, heuristics, neural networks, explainable AI

1. Introduction

Like any other area of science, there are unanswered questions in the field of constraint satisfaction search. In contrast to areas with a more marked empirical bent, often these questions receive little attention by researchers in the field. Apart from not being a focus of interest, it is typically not clear how to even approach such problems in order to gain a proper understanding of them. In addition, the solution to such problems is likely to be complex, which is of course why these questions remain unanswered. (Perhaps it is also because widely used theoretical frameworks such as complexity theory do not speak directly to such problems.)

The present project is concerned with one such problem, or unanswered question. This concerns a finding made some time ago in connection with a factor analytic study of variable ordering heuristics used in constraint satisfaction search [1, 2]. The basic measure used in the factor analysis was the number of search nodes in the search tree. This analysis showed that two classes of heuristics can be distinguished on the the basis of their loadings on one or the other of two factors. When proper controls were instituted, these two factors accounted for up to 94% of the variance. Most heuristics had high loadings on one factor and moderate loadings on the other, although certain advanced heuristics loaded highly on both factors.

All this suggested that two distinct kinds of activity were associated with heuristic selections; these were referred to as *heuristic actions*. Given the features used by heuristics that loaded most heavily on one or the other factor (e.g. degree of a variable versus current domain size) and certain properties of search trees generated when these heuristics were used [2], they were tentatively labeled as contention and simplification heuristics, and this was also the name given to the heuristic actions. However, at the time when this proposal was made, there was no clear understanding of the nature of these actions.

Recent work has helped clarify the nature of these heuristic actions ([3] and submitted and ongoing work). Simplification appears to work by separating the unassigned variables in the problem into different clusters with limited or no interaction among them. Contention uses information drawn from features such as domain reduction (the number of values discard from the set of possible assignments

Proceedings of the Workshops and Doctoral Consortium of the 42nd International Conference on Logic Programming (ICLP 2026), co-located with the Federated Logic Conference (FLoC 2026), July 18-19 and 24-25, 2026, Lisbon, Portugal.

r.wallace@ucc.ie (R. J. Wallace)

0000-0002-3537-7719 (R. J. Wallace)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC by 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-03

CEUR-WS.org/Vol-4311/RCRA_regular_04.pdf

to a given variable) or summed failures (wipe-outs) associated with a given constraint, in order to select places of contention within a problem for immediate assignment; this alleviates thrashing, which is a well-known source of inefficiency.

These experiments were mostly done with randomly generated problems with homogeneous connectivity and patterns of support, since heterogeneity and specialized structure make such an analysis more complicated and may well obscure any fundamental differences while reflecting fairly trivial ones. [1] Given that so many extraneous features were controlled for, it was conjectured that these two actions are the fundamental actions that occur as a result of successive instantiation of variables in the course of search. Heuristics work by inducing variable selections that enhance one or both of these actions.

Since the factor analysis revealed two distinguishable patterns of variation in search performance across problems, under the given hypothesis there must be differences among problems with respect to their amenability to the different forms of heuristic action. This implies that there must be different kinds of problem structure within what appears to be a homogeneous set of random problems that account for this difference in search efficiency. Until now, the nature of these differences has remained obscure, although it would be interesting and potentially useful to identify them.

Since machine learning methods have proven so powerful for recognizing patterns of features that enable them to classify objects belonging to different categories, it may be possible to use these methods to gain some insight into the basis for differences between problems in the situation just described. Therefore, the first step in this analysis will be to discover a set of problem features that allows a system such as a deep neural network (DNN) to classify problems of the sort used in the earlier experiments on the basis of whether they favor either the contention or the simplification action with respect to search efficiency. Given a highly accurate classification, then XAI techniques can be used to determine which problem features are most important. Although this will not be sufficient to give us a complete understanding of how specific problem features give rise to differences in search effort under different patterns of heuristic actions, it may indicate the features that will have to be taken account of in any model of these effects.

The present work is an initial step in this direction. In addition to trying to provide indications of the basis for critical differences among problems, there are a host of methodological issues that must be settled or at least clarified, which the present work addresses.

The contents of this paper are as follows. The next section reviews some basic ideas in the fields of constraint satisfaction, neural networks, and explainable AI that formed the basis for the present analysis. Section 3 lays out experimental methods. Section 4 describes the basic results, which are discussed in Section 5. Section 6 gives some general conclusions.

2. Background

2.1. CSPs

Here, we review some of the basic concepts in the field that form the background to the present work.

A constraint satisfaction problem (CSP) involves assigning values to a set of variables subject to restrictions on the way that values can go together. More formally, a CSP can be defined as a tuple, (X, D, C) where:

X is a set of *variables*, $X_1, \ldots, X_n$,

D is a set of *domains*, D_i, where each D_i is the set of *values* that may be assigned to variable X_i

C is a set of *constraints*. Each C_i belonging to C consists of a relation R_i and a particular subset of the variables in X, called the *scope* of the constraint. R_i is based on the Cartesian product of the values of the domains of the variables in the scope.

[1] An example of this is found in [2], where with heterogeneous problems a more complicated factor structure was obtained despite all parts having the same kind of random structure as with homogeneous random problems and therefore being subject to the same heuristic effects.

A *solution* to a CSP is an assignment or mapping from variables to values, $A = \{(X_1, a), (X_2, b), ..., (X_k, x)\}$, that includes all variables ($k = n$) and satisfies all the constraint relations in C.

In the present paper we only consider backtracking methods for finding solutions to CSPs. These methods build partial solutions that are consistent, extending them with further assignments until a full solution is obtained. If at some point in this process the partial solution cannot be extended, then search 'backs up' by retracting assignments and replacing them with new ones before attempting to extend the partial solution.

Backtrack search is typically enhanced by interleaving local consistency with search decisions, to form "hybrid search algorithms". Typically, local consistency is done after each new value assignment in order to simplify the problem further before proceeding to the next decision. (Interestingly, the same two heuristic actions are apparently in effect whether or not one interleaves local consistency in this fashion, cf. [2].)

The efficiency of backtrack search is further enhanced by using heuristics (rules of thumb based on maximizing or minimizing some problem feature) to choose the next variable or value to add to the partial solution. A well-known heuristic (mindom) chooses a variable with the smallest domain size; another (domain over forward degree or d/fd) extends this idea by choosing the variable for which the value of the domain size divided by the number of adjacent variables among those still uninstantiated is minimized.

2.2. ML and XAI

Neural networks have been used successfully in a variety of problems that require the system to recognize differences among complex patterns of features. Since their general mode of operation has been well described elsewhere [4], it will not be reviewed here. The present work employed deep neural networks (DNNs) to classify CSP problems from an apparently homogeneous set using multiple problem features, as described below.

For XAI, the present work used SHAP (SHapley Additive exPlanations) to evaluate the contributions of different input features to accurate classification. As measures of relative importance of different input features in the ML setting, SHAP values are unique in having certain desirable mathematical properties, first worked out in the context of games and coalitions and later applied to the problem of evaluating the output of black-box systems such as neural networks [5, 6]. Despite their computationally intensive character, efficient software packages are available that allow this type of analysis even with fairly large data sets.

3. Experimental Methods

3.1. General plan of analysis

Table 1 shows results from a typical factor analysis. In this case, where the influence of extraneous sources of variance such as value selection were controlled for, the two factors shown accounted for 94% of the variance. Hence, it was concluded that a two-factor model was sufficient to account for basic differences in heuristic performance with these problems. In this particular experiment (and under the present interpretation), Factor 1 reflects the contention action and Factor 2 the simplification action.

The present analysis employed two of these heuristics, maximum forward degree (FD) and the FF2 heuristic of Smith and Grant [7]. FD chooses a variable that has the highest number of constraints with other unassigned variables. FF2 calculates an approximation to average tightness across all the constraints between a variable and other unassigned variables, by maximizing the quantity

$$(1 - (1 - p_2^m)^{d_i})^{m_i}$$

where p_2 is the original constraint tightness, m is the original domain size (assumed constant for all variables) m_i is the current domain size of variable v_i (the variable being considered for assignment), and

Table 1
Factor analysis of search performance using different heuristics

heuristic	Factor 1	Factor 2
dom	**0.852**	0.387
d/dg	**0.752**	**0.606**
d/fd	**0.749**	**0.622**
fd	0.461	**0.871**
dg*fd	0.484	**0.862**
egsm	0.459	**0.871**
FF2	**0.854**	0.473
FF3	**0.862**	0.461
FF4	**0.850**	0.493
prom	0.468	**0.768**
stdg	0.544	**0.807**

Based on (log) mean search nodes for set of 100 random problems; 50 runs per problem with random value selection. Loadings > 0.6 in bold.
(From [2], ©2007 IEEE)

d_i is the forward degree of variable v_i. These heuristics were chosen because they were roughly similar in performance on the problems used in the factor analysis, while (as one can see from Table 1) FD loads strongly on the simplification factor, and FF2 loads strongly on the contention factor. In addition, it is easy to demonstrate that there are problems in this problem class for which FD outperforms FF2 decisively and vice versa [2].

The basic strategy of the present work was to examine problems from the same population of problems that was used in the factor analysis, selecting approximately equal numbers of problems that favored one or the other heuristic decisively. These problems were then assessed with respect to a set of quantitative features to produce a CSV file which could be fed to a DNN.

Putting this another way, what we are doing here is selecting the two tails from the distribution of relative efficiency, where one tail is based on problems that strongly favor FD and the other tail on problems that strongly favor FF2. This is done in order to make the classification easier, and hopefully the NN weights more revealing.

In this case, if a DNN can classify the problems with a high degree of accuracy, this means that there is sufficient information in the given feature measurements to decide which class a problem falls into. If this outcome can be achieved, then XAI techniques can be used to determine which features in the chosen set contributed the most to the successful classification. This may at least narrow the search for the problem features that favor one or the other heuristic action.

It is essential for the reader to understand that the purpose of this exercise is not to build an accurate classifier (although that's certainly something that can be considered in the future). Instead, the hope is that accurate classification on a *crafted set* of problems will reveal what kinds of problem features make a given problem amenable to one or the other heuristic. (The fact that many, perhaps most, problems are not so discriminating in this respect is also an interesting fact, but is not the subject of the present endeavor.)

In this case, therefore, the usual validation and testing procedures are not as critical. All that is required is highly accurate performance on the given sample of problems. Granted, there is still (as always) the question of whether the conclusions drawn are truly general; one reason to believe that they are is the size of the sample, along with the degree of accuracy.

3.2. Problem features used in these experiments

Since this problem set is homogeneous with respect to standard parameters, i.e. number of variables, domain size, constraint graph density and constraint tightness, none of these features can be used in the present analysis. However, there are a number of other features that do distinguish these problems, making an analysis like this possible. The key question was: will they be sufficient for a machine learning procedure to achieve high accuracy in classifying them as favoring one or the other kind of heuristic.

These are the features in the full set, some of which are explicated in the text below:

- standard deviation of the distribution of variable degrees
- skew of the distribution of variable degrees
- the mean of the 2-neighbourhood of a variable
- standard deviation of the distribution of variables' 2-neighbourhoods
- skew of the distribution of variables' 2-neighbourhoods
- size of the cycle cutset based on removing variables of highest degree
- mean degree for the k highest-degree variables
- number of connected components in the graph formed by the high-degree variables
- size of largest connected component of the high-degree variable graph
- density of the subgraph formed by the high-degree variables
- "coverage" ratio of high-degree variables
- "overlap" in the coverage of the high-degree variables
- mean maximum satisfiability for values in domains of high-degree variables across all constraints in the subgraph based on these variables
- mean minimum satisfiability for values in domains of high-degree variables across all constraints in the subgraph based on these variables
- number of solutions

The 2-neighbourhood of a variable contains the neighbours (adjacent variables) of a variable together with the neighbours of those neighbours. More precisely, the 2-neighbourhood of variable X_i is the set of variables X_j for which there is a path of length ≤ 2 between X_i and X_j.

A cycle cutset of a (constraint) graph is a set of arcs which when removed remove all cycles from the graph. The cycle cutset is not unique. In the present work, a cycle cutset was obtained by removing the arcs of the highest-degree variable at each step in the process, breaking ties lexically.

In setting the size k of the set of highest-degree variables, various values were tested. In this work three values of k were used: 5, 7, and 10. In some cases, one of these values was used for k; in other cases all three were used simultaneously. In the latter case, this means that for all features based on the variables of highest degree, three values were computed, one for each value of k, and all were included in the CSV file.

Two methods of selecting high-degree variables were considered: (1) choose the k highest along with other variables that have the same degree as the smallest degree value for this set, (2) choose exactly k high-degree variables, selecting the lexically lowest numbered variable(s) if there are further variables of degree equal to the smallest degree in this set. In the present work, the second method was used.

"Coverage" and "overlap" in the list above refer to constraints between variables in the high-degree set and variables in the rest of the problem. "Coverage" is the number of non-high-degree variables having at least one constraint with variables in the high-degree set. The coverage ratio (the quantity actually used) is the proportion of non-high degree variables that have a constraint with at least one high-degree variable. "Overlap" is the ratio of all constraints between high-degree variables and non-high-degree variables to the number of variables in the coverage.

Two features in the list above refer to features of domain values rather than variables. The "satisfiability" of a domain value a with respect to a given constraint is the proportion of values in the adjacent domain that support that value, i.e. that appear in a least one tuple of the relation associated

with that constraint. (Average satisfiability for all values in a domain with respect to a constraint is, of course, the complement of the constraint tightness.) The maximum (minimum) satisfiability is the highest (lowest) satisfiability for any value in a domain with respect to one constraint. Averages of the maximum (minimum) satisfiability were taken across all variable-constraint pairs in the subgraph formed by the (designated) high-degree variables.

The last feature is the number of solutions. For these tests this was entered as a fraction of the maximum number of solutions found for problems in this sample; hence the value was in the range (0,1].

In tests with one value for the number of high-degree variables, a total of 14 or 15 feature measurements were included, depending on whether the number of solutions was included. When three values of k were used, 30 feature measurements were included, 24 of which pertained to the high-degree variables. In these tests, the number of solutions was not included. In addition subsets of features were tested, as indicated below.

3.3. Problem selection

As indicated before, the problems used in these experiments were randomly generated and had homogeneous characteristics with regard to number of variables, domain size, constraint graph density, and constraint tightness. In fact, they were from the same population of problems as those used in the original factor analysis.

Specifically, all problems had 50 variables and each domain had 10 values. Constraints were binary. The constraint graph density was 0.184 and the tightness of each constraint was 0.37 (i.e. for a given domain value a in variable X_i and a constraint C_{ij} between X_i and adjacent variable X_j, the likelihood that a given value b in the domain of X_j would support a was 0.63). All problems had solutions; this was done by filtering during problem generation (cf. below).

The search algorithm used to solve these problems is called maintained arc consistency (MAC) [8]. In this version of hybrid backtracking, following each new assignment, AC is re-established in the subproblem formed by the variables not yet given an assignment. If this AC fails, then the last value assigned is retracted; if there are no more values to test for the last variable chosen, then search backs up and tries another value for the next-to-last variable chosen. This process continues until either a complete set of consistent assignments is found (aka a solution) or there are no more values to test at the highest level, which means there are no solutions. (In these experiments search was for one solution.) This algorithm was implemented in Common Lisp, specifically Gnu CLisp version 2.49 running under Unix on a Dell Poweredge R410 (2.4 GHz).

The CSV file was based on 1000 problems that were generated and tested as follows. After a problem was generated and found to have a solution, it was solved 50 times with each of the two heuristics. In each separate run, each value assigned to a variable was chosen at random from the remaining set of untested values. Then if the average search effort (i.e. mean search-tree nodes across the 50 runs) favored FD by a factor of 2:1 or FF2 by a factor of 50%, the problem was considered a candidate for inclusion in the CSV set. (These thresholds were chosen after extensive preliminary work; they allowed a roughly 50:50 split between problems that favoured one or the other heuristic decisively without requiring weeks or months that would have been required to collect enough problems that favoured FF2 by a factor of 2:1.) One additional criterion was used during selection, which was that the current set of selected problems did not favor one or the other heuristic by more than a 55:45 ratio; this was done to insure that throughout generation the set of problems of each type remained close to the 50:50 ratio required for the full set.

3.4. DNN testing and subsequent XAI

Two DNN programs have been used to date in these experiments. Both are written in Python. Different programs were used as one means of verifying the generality of the results.

The first (“Program 1") was based on code that came with the ebook *Machine Learning Algorithms from Scratch* by Jason Brownlee. The second was based on code found in a demo on the Web (Your Data Teacher: "How to explain neural networks using SHAP"). In both cases, code was modified to suit the purposes of the present experiments.

Program 1 uses neural network code in the torch.nn package; Program 2 uses the sklearn.neural_network.MLPClassifier. Both programs were run under Windows-10 Pro on a Dell Optiplex 7020 with an Intel i5-4590 CPU (3.3 GHz).

As it is presently set up, Program 1 first divides the sample of 1000 problems into training and test sets, using a .74:.26 split. Then it performs 5-fold cross-validation testing using the training set, for each fold saving the best weights found across 1000 epochs. At the end of this phase, the best weights found across all folds are tested on all 1000 instances. After this, another training run is carried out using the full training set (with 3000 epochs), and accuracy is tested on the test set. Finally the same weights are evaluated on all 1000 instances. These last two accuracies are the ones reported in this paper and used in the subsequent SHAP analysis. Program 2 first trains on two-thirds of the problems and tests on the other third.

After extensive experimentation with various configurations, numbers of training runs (epochs) and backpropagation schedules (batch sizes), the following parameter values were used with Program 1 for the present experiments because they gave the best results in a sample of five or more runs. For tests with 30 features, the DNN had 9 input and hidden layers with a uniform width of 30; the output layer had a width of 2. For tests with 14 or 15 features, the DNN had 6 input and hidden layers with a uniform width of 30; the output layer had a width of 2. With Program 1, for tests with subsets of the 14 features, the 9-layer configuration was used.

Very different results were found with Program, which in the original only had one layer. Here, testing with 6 layers did not improve performance; hence, the original configuration was retained for the experiments.

As noted before, in the k-fold cross-validation tests, the number of epochs was 1,000, while in the final training the number was 3,000. In both cases the batch size was five.

For Program 1, in all layers except the last hidden layer, the activation function was ReLu; the softmax function was used between the last hidden layer and the output layer. In this work the binary classification was treated as a multiclass task. The cross-entropy function was used as the loss function.

The original version of Program 2 used a logistic activation function, and this was retained in the present tests due to difficulties encountered when ReLu was used. (Specifically, the program repeated hung during the SHAP phase.)

In all cases the XAI method used was SHAP (SHapley Additive exPlanations); this was implemented with the Python shap package. In Program 1 DeepExplainer was added to the original neural network code; since the original code of Program 2 used the KernelExplainer, this was retained. With Program 1, it was possible to alter a line of the shap package so that the binary classification was treated a multiclass problem. This allowed shap bar plots that showed the relative importance associated with each heuristic. This was attempted with Program 2.

In addition, the Python package XGBoost was used to obtain correlation matrices for different sets of features.

To demonstrate that having several hidden layers in the Program 1 DNN was necessary for achieving results of high accuracy, tests were run comparing accuracies across repeated runs with DNNs of 1, 3 and 6 layers. In these tests 14 features were used, based on a set of 7 high-degree variables. Results are shown in Table 2. Clearly, multi-layered networks gave markedly better performance than networks with only a few layers on the majority of runs.

3.5. Test set construction

With Program 1, in addition to the full set of 30 features, various smaller sets were tested. Most often, this was a 14-feature set, based on one value of k for the set of high-degree variables. In the latter case, $k = 7$, since this seemed to be the most successful single set of variables based on the overall ordering

Table 2
Accuracies with varying numbers of hidden layers (Program 1)

	1 layer	3 layers	6 layers
	final test set accuracies		
mean	75.6	77.2	85.6
	final full-set accuracies		
mean	77.8	83.5	93.6
median	77.5	83.5	95.3
lowest	71.0	72.0	45.4
highest	85.0	94.0	97.1
# 90%	0	6	29

Based on 30 runs with each configuration. Last line is number of runs giving 90% accuracy on full-set test.

in the SHAP analysis, although it was hard to distinguish from $k = 5$. In addition various subsets of this smaller set were tested, as described in the next section.

To date, Program 2 has only been tested with a set of 14 features. As with Program 1, these were based on one value of k for the high-degree set, but in this case rather than a single k the best performing feature across the three values of k was chosen. (Tests with this set have also been done with Program 1; the results were similar to tests based on feature selections with one k value.)

4. Results

With Program 1 it was possible to obtain runs in which the final accuracy was very high; these were the runs of most interest. For tests with the full 30 features, final full-set accuracies across the entire sample ranged between 96.5% and 97.5%, while the preceding test accuracies for the same runs ranged from 87.3% to 91.2%. For tests with 14 features, the final accuracies ranged from 96.1% to 97.1%. while the preceding test accuracies ranged from 86.5% to 91.2%. Although even the final performances weren't perfect, some misclassifications were perhaps to be expected, since sampling errors in the original problem selection process may have occurred, especially for FF2. At any rate, these results show that the information contained in these features is sufficient to allow the DNN to achieve a very high degree of accuracy on this sample. Hence, we should be able to begin to derive an adequate explanation for the difference in amenability to one or the other heuristic action using these selected features.

Figure 1 shows the SHAP values for 14 features for one run of Program 1 in which the final accuracy was 97.1%. As mentioned above and indicated by the axis labels, these features were drawn from the original set of 30, limiting high-degree variable features to those based on sets of size 7.

As can be seen, although all features receive some weighting, there are considerable differences. The two most important features pertain to variations in variable degree. The third pertains to the density of connections between the high-degree variables and those outside this set. The fourth again pertains to variations in variable degree.

Program 2 gave roughly similar results, as shown in Figure 2. In particular, the two most important features were the same and received the same proportional weighting.

In tests where a measure based on the number of solutions was included, this feature received almost no weighting, in contrast to all the others. (For that reason, it was not included in tests such as this one.)

The stability of the orderings was evaluated in various ways. Using the 14-feature set and Program 1, the average ranks across ten runs where the accuracy was 96.1% or greater were calculated. When these were compared with the ordering shown in Figure 1, the mean discrepancy was only 1.0, and for the top eight features it was either 0 or 0.5. In addition, the average rank over these ten runs was compared with averages for ten runs done with Program 2. When both sets of average ranks were ranked, the correlation between them, using Kendall's Tau, was 0.451; although not large, this value is statistically

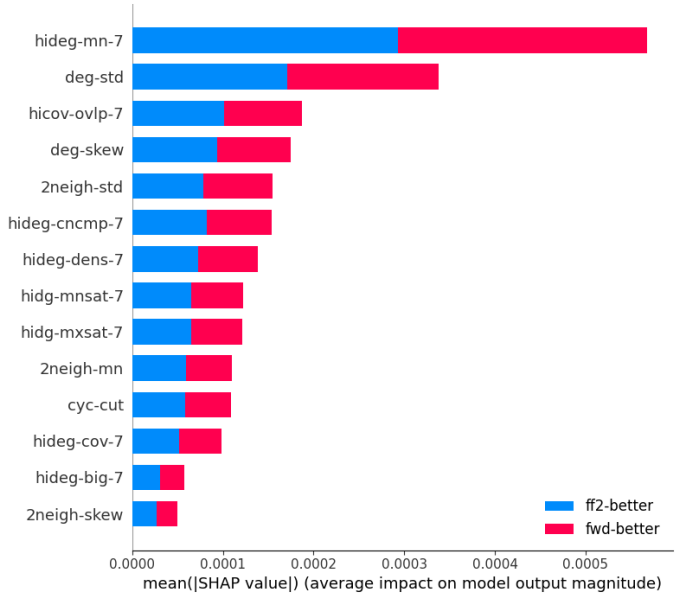


Figure 1: Mean SHAP values for weightings obtained with Program 1 when the final test accuracy was 91.2% and the final full-instance-set accuracy was 97.1%.

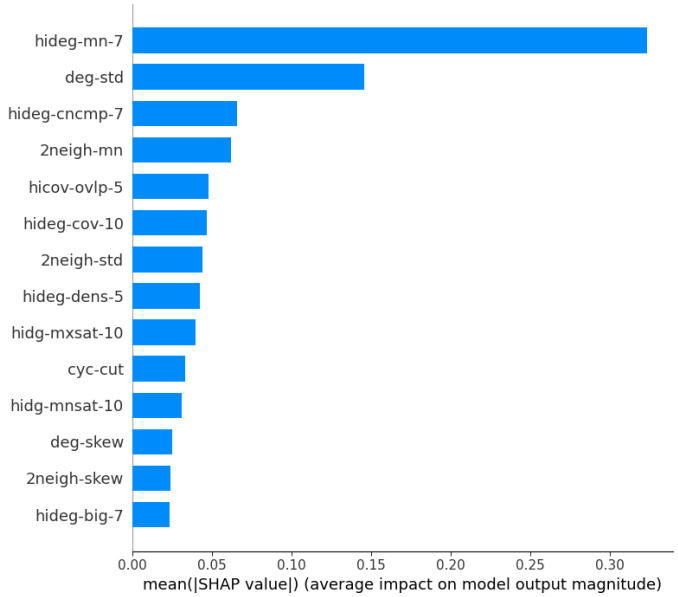


Figure 2: Mean SHAP values for weightings obtained with Program 2 when the final test accuracy was 83.9% and the final full-instance-set accuracy was 93.7%.

significant ($p \cong 0.0122$).

Further tests were done using Kendall's Coefficient of Concordance, W, for multiple sets of ranks [9, 10]. (W ranges from 1 (perfect concordance) to 0 (essentially random)) For the first set of ten runs (using Program 1) $W = 0.656$; transforming this to a Chi-squared statistic [10] gives a value of 85.3. For the set of ten runs done with Program 2, the same test gave a W of 0.830 and a Chi-square of 107.95. For both results, $p << 0.001$ for a null hypothesis of no concordance.

It appears, therefore, that reasonably stable results can be obtained with these methods, particularly for the most important features.

The Concordance analysis also provided another opportunity to evaluate the use of multiple layers in connection with Program 1. Concordances were obtained for the first 10 and best 10 runs (out of 30)

Table 3
Concordance with varying numbers of hidden layers (Program 1)

	first 10	best 10
1 layer	0.557	0.382
3 layers	0.495	0.451
6 layers	0.564	0.656

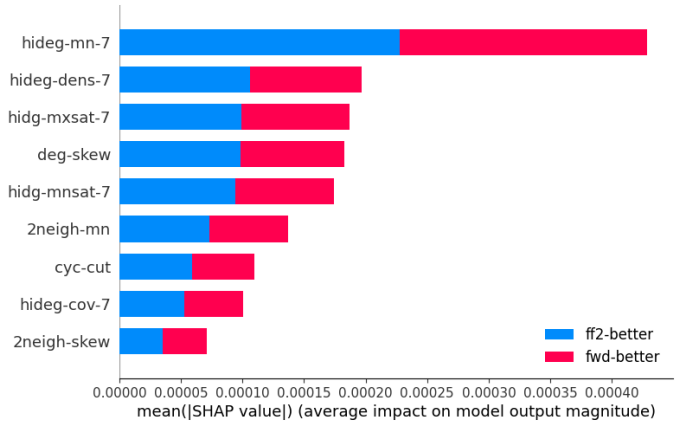


Figure 3: SHAP values for weightings for nine weakly correlated featues giving 96.3% accuracy.

for the 1-, 3-, and 6-layer neural networks. Results in 3 show that the reliability of SHAP results is also appreciably greater when a 6-layer network is used.

Tests of reliability with Program 2 gave even higher Concordances; thus for the first 10 runs using 14 high-degree features W = 0.759. Unfortunately, with this program the final full-set accuracies were 91.6-94.2. For this reason, most of the analysis was done with Program 1.

A correlation matrix associated with these 14 features plus a measure based on the performance ratio between forward degree and ff2 was obtained. Of interest is the fact that the correlations of single features with performance are never high, although there is a definite correspondence with the rank ordering of SHAP values. In particular, the four features with correlations whose absolute values are larger than 0.2 are those with the largest SHAP values (cf. Figure 1).

The question arises as to whether these four features are capable of giving high accuracy performance if used alone. This was tested by running Program 1 with a number of DNN configurations; in the best case, some runs gave full-set accuracies of 90-91%, while in the remaining cases accuracies were in the mid-80s or 70s. [2] Clearly then, information is required from the other features as well, in order to obtain high accuracies on this problem sample.

This was also verified by testing the DNN with the other ten features. (In these runs the same configuration was used as with the 4-feature set.) Surprisingly, DNN performance was as good if not better than with the four most important features, with the best runs yielding full-set accuracies of 95%.

Another series of tests was made where, beginning with the same 14-feature set, all correlations $\geq$ 0.4 were eliminated from the correlation matrix. This could be done by removing five features. In doing this, an attempt was made to retain those features that had higher correlations with the performance ratio. With this feature set, accuracies ranging from 92.4% to 96.3% were obtained, while 20% of the (15) runs 'failed', i.e. gave accuracies of around 50%.

A graph of SHAP values for the run giving 96.3% accuracy is shown in Figure 3. When the ranking

[2]It was also not uncommon to see 'bad' runs, with accuracies of around 50%, even with the best DNN configuration. This was extremely rare with the 14-feature set and was never seen with the 30-feature set over approximately 40 runs. Interestingly, such runs were not characterized by unusually low accuracies in the training phases.

of features on this run was compared to the rankings of average ranks over the best ten runs, the discrepancies ranged from 0 to 6. At the same time, appreciably smaller discrepancies were found when the average was based on a smaller range (here, 94.8% to 96.3%), suggesting again that with higher accuracies the relative importance of different features is fairly stable.

5. Discussion

Perhaps the most important conclusion that can be drawn from these results is that, although there may be only a few major features that determine differences in amenability to different heuristic actions, the full explanation may involve a number of features acting together in ways that are by no means obvious.

The most important features were all properties of the constraint graph, although the two micrograph features did receive moderate weightings. In hindsight, the importance of features like moments of the distribution of variable degrees is not surprising, although prior to this work the picture was entirely unclear. What could not have been anticipated was the significance of other properties, which in ways not yet understood also determine the relative effectiveness of different heuristic actions.

We can tentatively conclude that the existence of variables of very high degree is the critical feature that makes problems within this population more amenable to simplification than to contention. Clearly, if the maximum degree is higher, then the problem is likely to be split apart into more subproblems with reduced interaction between them. It can also be reasonaly assumed that contention is always significant, and when simplification effects are not sufficiently strong, then contention may become the most important heuristic action.

One rather surprising outcome was the finding that the number-of-solutions feature received no weight in the SHAP analysis, indicating that this feature apparently has little or no effect on the amenability of these problems to one or the other heuristic action. Before running these tests, it was thought that simplification might be the more effective strategy for problems with very few solutions, since it eventually produces many easy subproblems (a needle-in-the-haystack hypothesis). However, there was no evidence of such an effect in this study.

6. Conclusions and Prospects

Before proceeding, it may be worth discussing briefly why this work is of significance. In and of itself, an experiment on detailed characteristics of CSP search with random problems would not be of general interest. In the present case, however, the work is related to ideas that bear directly on a fundamental problem in this area, which is the nature of the variable ordering heuristics that play a critical role in effective search. In addition, they are directly related to a series of experiments that are currently the chief empirical basis for an explanatory framework that I would argue is potentially sufficient to account for heuristic performance. In this case, details matter, particularly those that are critical for interpreting these experiments.

The obvious limitation of this work is that it only provides qualitative hints in regard to the question it seeks to answer. And, of course, it is limited to problems with very little structure, especially in regard to constraints. Nonetheless, it does highlight structural features that may well be more general, in a problem class where their significance is not immediately obvious.

It is important to note that the present use of XAI techniques provides us, not just with hypotheses bearing on the question posed, but in addition, with empirical evidence regarding the problem features that are critical for differential performance with different heuristics, at least for this type of problem. In other words, they have not only helped explain, but have also served to verify. In addition, it is important to note that the pattern of results found in these experiments is consistent with and therefore provides additional evidence for the hypothesis that there are two basic heuristic actions.

In this case, the basic logic or inferencing seems sound; the only problem lies in the vagaries of the weighting process that occurs in training with the DNN. This has been addressed both by using

methods designed to steer the weighting process away from idiosyncratic trajectories and by statistical analyses based on repeated runs beginning with randomly chosen weights. This has been successful in producing reasonably stable results.

Future work will involve a more systematic analysis of subsets of the original feature set, especially when high correlations between different features are eliminated. Beyond that, there is obviously much to do to bridge the gap between SHAP weightings and a full-fledged model of the relation between problem features and performance, not to mention the extension of any such framework to the vast menagerie of problems that can be viewed as CSPs.

7. Acknowledgements

This publication has emanated from research supported in part by a research grant from Science Foundation Ireland under Grant No 12/RC/2289_P2. Table 1 was first published in the Nineteenth International Conference on Tools with Artificial Intelligence - ICTAI 2007, published by IEEE Press.

8. Declaration on Generative AI

No generative AI tools were used in preparing this paper.

References

[1] R. J. Wallace, Factor analytic studies of CSP heuristics, in: Principles and Practice of Constraint Programming-CP'05. LNCS No. 3709, Springer, 2005, pp. 712–726.

[2] R. J. Wallace, Determining the basis for performance variation in CSP heuristics, in: Nineteenth International Conference on Tools with Artificial Intelligence - ICTAI'07, IEEE Press, 2007, pp. 473–480.

[3] R. J. Wallace, Information flows during constraint satisfaction search with dynamic heuristics: information from preprocessing, in: M. Maratea, L. Serafini, M. Vallati (Eds.), 31st RCRA Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion, AI4-IPS-RCRA-SPIRIT 2024, CEUR Workshop Proceedings, Vol-3883, 2024.

[4] C. C. Aggarwal, Neural Networks and Deep Learning, 2 ed., Springer, 2023.

[5] E. Štrumbelj, I. Kononenko, M. R. Šikonja, Explaining instance classifications with interactions of subsets of feature values, Data and Knowledge Engineering 68 (2009) 886–904.

[6] C. Molnar, Interpretable Machine Learning, Leanpub, 2019.

[7] B. M. Smith, S. A. Grant, Trying harder to fail first, in: Thirteenth European Conference on Artificial Intelligence-ECAI'98, Wiley, 1998, pp. 249–253.

[8] D. Sabin, E. C. Freuder, Contradicting conventional wisdom in constraint satisfaction, in: A. Borning (Ed.), Principles and Practice of Constraint Programming - PPCP'94. LNCS No. 874, Springer, 1994, pp. 10–20.

[9] S. Siegel, Nonparametric Statistics for the Behavioral Sciences, McGraw-Hill, New York, 1956.

[10] Anonymous, Kendall's W, in: Wikipedia, 2026.