Combining Variable Ordering Heuristics for Improving Search Algorithms Performance

Abdolreza Hatamlou   Yusef Farhang   Mohammad Reza Meybodi

***Abstract*—**Variable ordering heuristics are used in constraint satisfaction algorithms. Different characteristics of various variable ordering heuristics are complementary. Therefore we have tried to get the advantages of all heuristics to improve search algorithms performance for solving constraint satisfaction problems. This paper considers combinations based on products and quotients, and then a newer form of combination based on weighted sums of ratings from a set of base heuristics, some of which result in definite improvements in performance.

***Keywords*—**Constraint Satisfaction Problems, Variable Ordering Heuristics, Combination, Search Algorithms.

I. Introduction

MANY problems in the fields of artificial intelligence, network, database, engineering and other areas of computer science can be viewed as special cases of constraint satisfaction problems, including image processing, natural language parsing, routing, circuit design, scheduling and more. A CSP is a problem composed of a finite set of variables, each of which is associated with a finite domain, and a set of constraints that restricts the values the variables can simultaneously take [1]. The search space of CSPs is often exponential. Therefore a number of different approaches to the problem have been proposed to reduce the search space and find a feasible solution in a reasonable time. There are many variable ordering heuristics that can improve the search algorithm performance. There are listed below together with the abbreviations used in the rest of paper [1], [3].

- Minimum domain size (dom, dm). Choose a variable with the smallest current domain size.
- Maximum forward degree (fd). Choose a variable with the largest number of neighbors (variables whose nodes are adjacent in the constraint graph) within the set of uninstantiated variables.
- Maximum backward degree (bkd). Choose the variable with largest number of neighbors in the set of instantiated variables.
- Maximum static degree (stdeg, dg). Choose a variable with the largest number of adjacent variables in the constraint graph (variable of highest degree)

Combining variable ordering heuristics that are based on different features sometimes results in better performance that can be obtained by either heuristic working in isolation. Perhaps the best-known instance of this is the domain/degree heuristic. Recently, further examples have been found based on weighted sums of rated selections produced by a set of heuristics.

As yet, we do not have a good understanding of the basis for such heuristic synergies. Nor can we predict in general which heuristics will synergize. In fact, until now there has been no proper study of this phenomena, and perhaps not even a proper recognition that it is a phenomenon. The present paper initiates a study of heuristic synergies. A secondary purpose is to test the weighted sum strategy in a setting that is independent from its original machine learning context.

The tests in this paper were based on the MAC-3 algorithm with random constraint satisfaction problems [1], [2], [4]. The main tests were based on: node visited during search.

Synergy was evaluated for two kinds of strategy. The first type of strategy was to take products and quotients of the basic heuristics, as is done in the well-known domain/degree heuristics. The second was to combine evaluations of individual heuristics into weighted sums.

Weighted sums were obtained by assigning each individual heuristic a weight, then at each choice point allowing all heuristics to rate the choices (variables) beginning with a score of 10 as the best rating and 9 for the next-best rating, 8 for the next, and so forth down to a minimum of 1. For example, if the current domain sizes were 1, 2, 2, 3, 4, 4, 5, the corresponding ratings for min domain would be 10, 9, 9, 8, 7, 7, 6. The ratings were then combined for each choice by multiplying the rating for each heuristic by its weight and adding the products. Then, the variable with the highest weight was chosen (again, ties were broken lexically, by choosing the variable with the smallest integer label).

II. Heuristic Combinations

A. Based on Quotients and Products

Aside from the domain/degree heuristics, it was not possible to find quotients or products that gave clearly superior performance. Interestingly, for this set of problems

This work was supported by Islamic Azad University – Khoy branch.

Abdolreza Hatamlou is with Islamic Azad University, Khoy 58135-175, Iran (phone: +98 914 462 7893; e-mail: rezahatamloo@gmail.com).
Yusef Farhang is with Islamic Azad University, Khoy 58135-175, Iran (phone: +98 914 461 0199; e-mail: yfarhang@yahoo.com).
Mohammad Reza Meybodi is with the Computer Engineering Department, Amirkabir University of Technology Tehran 15875-4413, Iran (e-mail: mmeybodi@aut.ac.ir).

domain/static degree did not give better performance than static degree alone, while domain/forward degree was clearly superior to its component heuristics. (Note that for heuristic combinations involving backward degree, when this component was zero, 1 was used instead in both the quotients and products.)

TABLE I
RESULTS FOR PRODUCTS AND QUOTIENTS

SIMPLE HEURISTIC	NODES	COMBINATION	NODES
DOM	11,554	MIN DOM/STDEG	2296
FD	2845	MIN DOM/FD	**1841**
BKD	27611	MIN DOM/BKWD	15301
STDEG	2220	MAX FD/BKWD	3106
		MAX STDEG * FD	2637
		MAX BKWD * FD	2667

Mean nodes per problem. < 50,10,0.184,0.32> problems. Bold entries show results that are better than any individual Heuristic.

B. *Based on Weighted Sums*

Several demonstrations of heuristic synergy, as well as a lack thereof, are shown in Table II. This gives results, in terms of nodes searched, for five individual heuristics and for "combinations" of these heuristics using the technique of weighted sums. For these tests, heuristics were given equal weights.

Note that in these tests the domain/degree quotients were also used as components with respect to the weighted sums, and that this sometimes gave better results than the quotient alone. On the other hand, it was never clearly superior to the best combinations based on simple heuristics.

The important things to note regarding these results are:

- Some combinations do better, in terms of number of search nodes, than any heuristic used by itself. Some do even better than the best heuristic-quotient tested, which was min domain/forward-degree.
- There is considerable variation in search efficiency with different combinations. In some cases there is no improvement over the best heuristic in the mix; in some there is marked improvement.
- The best results for combinations of two heuristics were as good as the best results for combinations of more than two heuristics.

In addition, weighted sums gave better results than tie-breaking strategies based on the same heuristics. For comparison, here are results on the same set of problems with four tie-breaking strategies:

- min domain, ties broken by max forward degree: 3321
- min domain, ties broken by max static degree: 3375
- forward degree, ties broken by min domain: 2459
- static degree, ties broken by min domain: 1826

Note that, as expected, tie-breaking does reduce the size of the search tree in comparison with the primary heuristic when used alone.

TABLE II
RESULTS FOR WEIGHTED SUMS

HEURISTIC	NODES	COMBINATION	NODES
DOM	11,554	DOM+DM/DG	2547
DM/DG	2296	DOM+DM/FD	**1537**
DM/FD	1841	DOM+FD	**1537**
FD	2845	DOM+BKWD	12741
BKD	27,611	DOM+STDEG	**1647**
STDEG	2220	DM/DG+DM/FD	2020
		DM/FD+FD	**1524**
		DM/FD+STDEG	**1606**
		FD+STDEG	2564
		DM/FD+BKWD	2081
		BKWD+STDEG	2096
		DOM+DM/DG+STDEG	1874
		DOM+FD+STDEG	**1594**
		DOM+FD+BKWD	**1650**
		DOM+BKWD+STDEG	2042
		FD+BKWD+STDEG	2211
		DOM+DM/DG+FD+BKWD	1877
		DOM+DM/DG+FD+STDEG	**1594**
		DOM+DM/DG+BKWD+STDEG	2054
		DOM+DM/DG+FD+BKD+STDEG	**1690**

Mean nodes per problem. < 50,10,0.184,0.32> problems. Bold entries show results that are better than any individual Heuristic.

Another significant result is that, for the best combinations of two heuristics, equal weights gave better results than unequal weights, and in these cases performance deteriorated as a function of the difference in weights. This is shown in Table III. In cases in which weight combinations did not 'synergize' or synergized weakly in comparison with the best individual heuristic in the combination, unequal weights sometimes gave some improvement, although the effect was never marked.

TABLE III
TWO-HEURISTIC COMBINATIONS WITH DIFFERENT WEIGHTS

WT PATTERN	DOM+FD	DOM+STDEG	STDEG+BKWD	STDEG+FD
1:1	1317	1427	1876	2344
1:2	1433	**1420**	2471	2455
2:1	1405	1620	**1852**	**2235**
1:3	1652	1454	3054	2458
3:1	1651	1885	**1812**	**2223**
1:5	2033	1557	3960	2458
5:1	2368	2504	**1816**	**2223**

Mean nodes per problem. <50,10,0.184,0.32> problems. The weight pattern, read from left to right, corresponds to the weights given to each heuristic each combination, again reading from left to right. For ease of comparison, the results for equal weights are repeated in the first row of the table. Bold entries show results that are better than the 1:1 condition.

III. CONCLUSION

This paper presents a study of the effects of combining heuristics. The most interesting direct result was that weighted sums are in fact quite good at improving search performance in terms of search nodes; this must be because this strategy improves the quality of variable selection. We believe that these heuristics can be successfully applied to many constraint satisfaction problems and other algorithms for solving CSPs.

REFERENCES

[1] E. Tsang. Foundations of Constraint Satisfaction. Academic Press, 1993.
[2] K. Marriot, P. J. Stuckey. Programming with Constraints: An Introduction. The MIT Press, 1998.
[3] Z. Michalewicz and D. B. Fogel. How to Solve It: Modern Heuristics. Springer-Verlag, 2000.
[4] C Stefan Vo?. Meta-heuristics: State of the art. In Alexander Nareyek, editor, Lcal search for planning and scheduling: revisited papers, pages 1–23. Springer-Verlag LNCS 2148, 2001.