Exploiting Problem Structure as a Search Heuristic

Tad Hogg

Intl. J. of Modern Physics C **9**, 13–29 (1998)

Abstract

Recent empirical and theoretical studies have shown that simple parameters characterizing constraint satisfaction problems predict whether they have a solution and the cost to solve them, on average. This paper examines the effectiveness of using these predictions as a heuristic for solving the graph coloring problem. Specifically, by adding some global information on the consequences of various choices, the use of these parameters can reduce the search required to find a solution. Current limitations of this approach, due to the high variance associated with the predictions, are also presented. More generally, observations of universal behaviors analogous to physical phase transitions can be applied to improve search methods.

1 Introduction

Combinatorial search problems have easily computable characteristics that determine, on average, both the likelihood they have solutions and their *hardness*, i.e., the cost to solve them with a variety of heuristic search methods [3, 20, 5, 30, 16, 28, 25, 12]. Important recent observations are abrupt transitions in behavior, analogous to physical phase transitions in percolation problems [26]. In particular, these transitions are universal characteristics of classes of hard computational problems, independent of the choice of particular search algorithms.

These results provide insight into the nature of NP-hard problems, but can they also lead to improved search methods? On the one hand, this seems likely in that all heuristics rely on approximate estimates of solubility and search cost associated with various choices to be made during the search. Thus the use of universal problem characteristics may usefully add global information to heuristics that evaluate choices based on limited, local views of the problem. However, the relation between these characteristics, solubility and hardness only holds on average: the large observed variances indicate that any individual problem instance can deviate significantly from the average behavior. This variance could easily render useless any reliance on average behavior.

This issue has been successfully resolved for some search problems using genetic algorithms [4]. Genetic algorithms provide an appealing basis for using knowledge of the average problem behavior as a heuristic because they are a statistically based search method relying on improvements to a population on average. Thus they are likely to be less sensitive to the variance associated with the heuristic than, say, a conventional backtrack search method. More recently this improved understanding of the likely difficulty of search problems has been applied to create a simple domain-independent heuristic [9].

This paper continues this line of work by evaluating the use of problem hardness for a complete search method based on depth-first backtracking. This involves extending previously developed theories to apply to the situations encountered during such a search, and making specific choices on how to use the global information provided by the theory. The key insight allowing the theory to apply is that, at any point during a backtrack search, the remaining unassigned variables and the constraints on them constitute a subproblem of the original one. The theory can then be applied to this subproblem to suggest improvements in subsequent choices. The results are evaluated empirically for the particular search problem of graph coloring.

2 Problem Characteristics

In this section, we relate solubility and search cost to easily computed problem characteristics. This gives an abrupt transition from underconstrained problems that are mostly soluble to overconstrained ones that are mostly insoluble as a parameter is varied. This behavior and its scaling properties are analogous to phase transitions seen in many physical systems [14, 16]. In terms of search cost, the hardest cases are concentrated near this transition for a variety of search methods. For use as a heuristic, the most accurate approach for a given class of constraint problems is to make use of known empirical results for the location of the transition point. However, computationally this is cumbersome for use as a general-purpose method, and in many cases the required empirical results are not available, or the actual search problems could differ significantly from the randomly selected ones used in most of these studies. Hence we consider instead an approximate theory [30] which can be readily applied, but does not exactly determine the transition points in particular cases. Although this would appear to reduce the accuracy of the results, we should note that real problems, and especially those subproblems encountered during a search, may differ from the classes of random problems considered in these studies. This is likely to make more of a difference than the relatively small error in the location of the transition point introduced by using the theory instead of exact results, and thus provides additional motivation for using the readily computed theoretical values rather than relying on a table of empirical values[1].

2.1 Hardness theory

A constraint satisfaction problem (CSP) [18, 17] consists of a set of n variables, a domain of values for each variable, and a set of constraints each of which restricts the allowable assignments of values to some of the variables. Let b_i be the number of values for variable i. The task is to find a solution, i.e., an assignment of a value to each variable such that all constraints are satisfied, or else to establish that no such solution exists. We define a *state* as a set of assignments, not necessarily consistent with all the constraints, to some of the variables. Theoretically, it is sufficient to characterize the constraints by the number m and size k of the *minimized nogoods*. These nogoods are simply those smallest subsets of all possible states in the problem that violate at least one constraint, and are readily determined from the nature of the constraints. The *size* of a nogood is just the number of variables it gives an assignment to. For simplicity we consider the case of binary constraints, for which each of the nogoods involves assignments to exactly two variables (i.e., $k = 2$). This is the relevant case for the problem of graph coloring considered below. Cases with constraints involving more variables (such as 3-SAT used in many empirical studies of the transition behavior), as well as different-sized constraints, can be handled as a generalization of this theory [9, 29].

A simple example is a problem with three variables, x, y and z, with 2, 3 and 4 values respectively, and the constraint that each pair has distinct values. In this case, the minimized nogoods are all assignments of the same values to a pair of variables. E.g., those involving the variable x are $\{x = 1, y = 1\}$, $\{x = 1, z = 1\}$, $\{x = 2, y = 2\}$ and $\{x = 2, z = 2\}$. The remaining nogoods involve y and z, namely $\{y = 1, z = 1\}$, $\{y = 2, z = 2\}$ and $\{y = 3, z = 3\}$.

Theoretically, the search behavior is determined by the expected number of consistent states with various numbers of assigned variables. To allow for the variables having different domain sizes, we present an extension to the previously developed theory [30]. First, note that there are $b_i b_j$ possible minimized nogoods for variables i and j, for a total of $N = \sum b_i b_j$ where the sum is over all distinct pairs of variables. Equivalently $N = \frac{1}{2}\left(\left(\sum b_i\right)^2 - \sum b_i^2\right)$, with the sums over all variables. A given state with assignments to j variables will satisfy all constraints provided none of the value pairs specified in the state are in the set of nogoods for the problem. Assuming random selection of the minimized nogoods, the state is consistent with probability

$$p_j = \frac{\binom{N-\binom{j}{2}}{m}}{\binom{N}{m}} \tag{1}$$

This is simply the ratio of the number of ways the m nogoods can be selected so that the given state is consistent to the total number of ways the nogoods can be selected. For the class of problems we consider

[1]For the graph coloring problems considered here, the theory predicts the location of the transition point to within 10–20%, and can be improved further [29].

below, generally N will grow as $O(n^2)$ while m grows as $O(n)$, which gives the asymptotic behavior

$$\ln p_j \sim m \ln \left(1 - \frac{\binom{j}{2}}{N}\right) \tag{2}$$

Considering a specific ordering on the variables, the expected number of consistent states, or *goods*, with assignments to the first j variables is

$$G_j = S_j p_j \tag{3}$$

where

$$S_j = \prod_{i=1}^{j} b_i \tag{4}$$

is the total number of possible assignments to the first j variables in the given ordering. In particular, the expected number of solutions is $N_{\text{soln}} = G_n$. When there are few nogoods (small m), the expected number of solutions grows exponentially with n. Conversely, when there are many nogoods, it decreases exponentially. Between these extremes is a transition from mostly soluble to mostly insoluble cases whose location is estimated by the value of m for which $N_{\text{soln}} = 1$.

The cost for a simple nonheuristic backtrack search is then estimated as [30]

$$C_{\text{1st}} = \frac{C}{\max(1, N_{\text{soln}})} \tag{5}$$

where $C = \sum G_j$ corresponds to the search cost to find *all* solutions. This is readily understood as follows. Finding all solutions through a backtracking search starts with the empty assignment, and successively attempts to assign values to additional variables in the specified ordering as long as the assignment is consistent with the constraints. When any inconsistency is found, the search procedure backtracks to a previous decision point and tries another value. In the most direct form of backtracking, this procedure will consider all consistent states in the process of finding all the solutions, leading to the expression of C. While more sophisticated search methods can reduce the number of states examined, the relative qualitative behavior determined from this simple case remains the same. With this estimate for the cost to find all solutions, we obtain an estimate of the average cost to find the first solution, if any, by assuming the solutions are randomly located in the search space, giving the expression for C_{1st}.

Qualitatively, C_{1st} increases as nogoods are added, reaches a maximum at the predicted transition from soluble to insoluble problems, and then decreases. For large n the terms in the sum for C grow rapidly up to some maximum value then decline. Thus the behavior of C can be approximated by the largest term in the sum. This cost estimate involves more approximations than used to determine the location of the transition point, and moreover is associated with a large variance so even accurate knowledge of the *mean* cost value of limited use in actual searches.

Having different domain sizes introduces a new complication in the theory, namely in what order one should consider the variables when forming the product used in Eq. (4). One choice is to consider first the variables with the smallest value of b_i, and then those with successively larger values. This corresponds to the heuristic used in the experiments reported below, and is a reasonable choice at least in extreme cases. For instance, if any variable has $b_i = 0$ the problem clearly has no solution and such variables should be considered first. Then any variable with $b_i = 1$ has its choice forced and should be made immediately to allow possible pruning of further choices as soon as possible (this amounts in effect to a forward propagation during the search, a method that can be quite helpful in practice [24]).

2.2 An example

To make this theory more concrete, consider the case in which all variables have either 1, 2 or 3 possible values. This corresponds, for example, to the case of 3-coloring considered below. Let n_r of the n variables have $b_i = r$, for $r = 1, 2, 3$. In this case, $N = \frac{1}{2}\left(\left(\sum n_r r\right)^2 - \sum n_r r^2\right)$. Ordering the variables according to their value of b_i then gives

$$S_j = \begin{cases} 1 & \text{if } j \le n_1 \\ 2^{j-n_1} & \text{if } n_1 < j \le n_1 + n_2 \\ 2^{n_2} 3^{j-n_1-n_2} & \text{if } n_1 + n_2 < j \end{cases} \tag{6}$$

To obtain a simple explicit form for the cost proxy, we can make the additional approximation to Eq. (2): $\ln p_j \sim -m\frac{\binom{j}{2}}{N} \sim -\frac{mj^2}{2N}$ and hence $\ln G_j \sim \ln S_j - \frac{mj^2}{2N}$. The main contribution to the overall cost C is obtained by maximizing[2] $\ln G_j$. With this approximation, $\ln N_{\text{soln}} \sim \ln S_n - \frac{mn^2}{2N}$ which can be combined with the maximum of $\ln G_j$ to give an explicit estimate for the cost $\ln C_{\text{1st}}$.

3 Improving Heuristics with Global Information

In this section, we describe the general backtrack search method and the decision points at which heuristics can guide choices. We then discuss how global knowledge of whether problems are soluble and their associated search cost can be used.

3.1 Backtrack search

There are a large variety of backtrack search methods for CSPs [24]. These are *complete*, i.e., guaranteed to correctly determine whether a solution exists and, if so, find a solution. Specifically, the depth-first chronological backtrack procedure backtrack(*state*,d) operates on a state consisting of a consistent assignment to d of the variables (also called a partial solution). The procedure can be described as follows:

```
if ($d$ equals $n$) then
  return state as a solution
else
  pick unassigned variable $v$ in state
  for each $i$ that is consistent for $v$ do
      $s$ = state with $v$ assigned value $i$
      result = backtrack($s$, $d+1$)
      if (result is a solution) then
        return result as a solution
      endif
  endfor
  return "no solution"
endif
```

The search begins with a completely unassigned state and with d set to zero.

3.2 Subproblems that arise during search

At each point in the search, the unassigned variables constitute a subproblem remaining to be solved, in which any values in their domains that are inconsistent with assignments already made are simply excluded from consideration.

As an example, the subproblem generated by assigning a value α to the first variable, i.e., $v_1 = \alpha$, consists of

- the $n-1$ remaining variables,
- the sets of allowed values for each of these variables that are consistent with the new assignment, i.e., any nogood of the form $\{v_1 = \alpha, v_i = \beta\}$ would remove the value β from the set of possibilities for variable i,
- the set of minimized nogoods that do not involve the assigned variable, and whose assignments are allowable[3]

[2]The maximum can be found explicitly since $\ln G_j$ is piecewise quadratic, and is thus simpler than the transcendental equation for the maximum in the original theory [30].

[3]If the problem has constraints among more than two variables, the minimized nogoods of the subproblem would also include those involving $v_1 = \alpha$ restricted to the remaining variables. E.g., $\{v_1 = \alpha, v_2 = \beta, v_3 = \gamma\}$ would become $\{v_2 = \beta, v_3 = \gamma\}$. This could give rise to a subproblem with nogoods of different sizes. Note also that such nogoods arising from different constraints could be identical hence requiring a check to avoid multiple counts of the same nogood for the subproblem.

An example is the CSP of coloring a triangle with 3 colors such that each vertex has a distinct color. Initially we have the values (i.e., colors) $\{1, 2, 3\}$ for each of the three variables (i.e., vertices of the triangle), together with the minimized nogoods preventing any pair from having the same color, i.e., $\{v_1 = c, v_2 = c\}$ for $c = 1, 2, 3$ and similarly for the other pairs of variables. Suppose we make the assignment $v_1 = 1$. The resulting subproblem has variables $\{v_2, v_3\}$, each with domain $\{2, 3\}$ and the remaining nogoods $\{v_2 = c, v_3 = c\}$ for $c = 2, 3$.

3.3 Using heuristics

Backtracking has two natural points for using heuristics. First is selecting the next unassigned variable to consider. Ideally, if the previously assigned values preclude any solution, one should select next the variable that most rapidly leads to a conflict thereby minimizing the amount of search required before backtracking. On the other hand, if this search path does lead to a solution, all variables will need to be considered eventually anyway so nothing is lost by selecting the variable most likely to give a conflict. A simple, and often effective, way to do this is to pick next the variable with the fewest remaining available assignments.

The second use for a heuristic is in the order of assigning values to the new variable. Here we would like to first try values most likely to lead to a solution, if any, consistent with the current partial assignment. If successful, this eliminates the need to consider additional values for the new variable. Here a useful method is to select the least constraining values first in the hope that they will leave open many options for the remaining unassigned variables.

These two decision points complement each other. If value selection is done well and the problem is soluble, there will be few choices leading to deadends and backtrack. Thus it will be relatively less important for the overall search cost that the occasional incorrect choices are pruned quickly by good variable ordering. Conversely, if the value selection has many errors, or the problem is not soluble, then variable ordering will be particularly important to limit the cost of the many incorrect choices by rapidly uncovering conflicts.

Conceptually, these heuristics could always correctly determine the best possible choice by complete search, i.e., solving the problem from the current choice point. But this would, of course, require as much computational effort as doing the search directly. So instead the heuristics are further restricted by the requirement that they execute rapidly. Thus, in practice, these heuristics evaluate potential choices by the use of local information only, i.e., how the choices directly affect other variables, but do not consider how those effects may subsequently affect future search and so will not always give the best possible choice. Hence it could be useful to supplement the local information used by the heuristic with easily computed estimates of global properties of the problem.

3.4 Using solubility and cost estimates

The appeal of using universal problem characteristics to determine solubility and search cost of the subproblems is that these global quantities, if estimated well enough, can usefully guide the choices made during backtrack.

Consider the order in which values should be examined for a newly selected variable v. Let $p(v = i)$ be the estimated probability that the subproblem remaining after v is given the value i is soluble. A simple heuristic would then be to consider the values in decreasing order of $p(v = i)$. More sophisticated choices could also consider the cost estimates: in some cases it may be worthwhile to first try a value with low cost even if its probability for success is somewhat lower than some other, higher cost, choice.

For variable ordering, we can attempt to select next the variable that will minimize the search cost. This is most important when the current partial assignment does not lead to a solution. Let $C(v = i)$ denote the cost estimate for the subproblem that remains after v is given the value i. If there is no solution, the total cost for selecting variable v next will be $C(v) = \sum_i C(v = i)$. We can then pick the variable that minimizes this value.

We thus see how cost and solubility information on the subproblems arising during search can be used as a heuristic. Note this does not require accurate absolute cost values, rather it is sufficient that the estimates closely preserve the ordering of the costs of different choices, not the actual values. This motivates using the easily computed cost of a simple search method given in §2 because, as observed in many studies, hard cases for a variety of search methods tend to be associated with the same values of problem structure parameters.

Moreover, when the costs are fairly close, even the correct ordering is not that important. Thus we are mainly interested in estimates of the values that are sufficient to avoid particularly high-cost choices.

3.5 A hardness-based heuristic

There remains the question of how to best add the global information approximately estimated by the theory to the local information already used by a search heuristic. At one extreme, the theory could be used to order the choices, using local information only if there are any ties. At the other extreme, the global information could be used only to break any ties that arise locally. The choice of either of these extremes, or some intermediate policy, will depend on how accurate the global information is relative to that obtained locally. Since the theoretical cost estimates have a large variance which results in lower accuracy for individual instances, a simple policy is as follows.

For ordering the values, we rely only on the solubility estimates provided by the theory. Specifically, values are assigned[4] in order of decreasing magnitude of p_n, given by Eq. (2). Any ties are resolved by the local heuristic.

Variable selection emphasizes choices that lead to early failure, i.e., subproblems with no solutions. In this case the solubility estimates of the theory are of no use and we must instead rely on the less accurate cost proxy. In our experiments, we used the cost proxy described in §2.2 only to break ties that arose from the local heuristic. In these cases we selected next the variable with the lowest estimated cost.

4 Graph Coloring

Our experiments used the graph coloring problem. This consists of a graph, a specified number of colors b, and the requirement to color each node in the graph so that no pair of adjacent nodes (i.e., nodes linked by an edge) have the same color. Many important artificial intelligence problems, such as planning and scheduling [31, 2], can be mapped onto the graph coloring problem. Moreover, as a well-known NP-complete problem, graph coloring has received considerable attention [19, 15, 27].

We use the Brelaz heuristic [15] which assigns the most constrained nodes first (i.e., those with the most distinctly colored neighbors), breaking ties by choosing nodes with the most uncolored neighbors (with any remaining ties broken randomly). The colors are considered in order $1, 2, \ldots, b$ and, for each node, the smallest color consistent with the previous assignments is chosen first, with successive choices made when the search is forced to backtrack. As a simple optimization, we never change the colorings for the first two nodes selected by this heuristic. Any such changes, which could only occur when the backtrack search has failed to find a solution starting from the initial assignments for the first two nodes, would amount to unnecessarily repeating the search with a permutation of the colors.

For this problem, the connectivity γ (i.e., the average degree of the graph) distinguishes relatively easy from harder problems, on average [3]. This is related to the number of edges e and number of nodes n in the graph by $e = \frac{1}{2}\gamma n$. Fig. 1 shows the peak in the median search cost at $\gamma = 4.6$ which is also where the fraction of graphs with a solution drops from near one to near zero. The peak and transition in solubility become sharper as larger graphs are considered.

5 Experiments

The heuristics described in §3.5 were implemented for graph coloring using the Brelaz method as the local search heuristic. We measure the search cost by the number of states examined until the first solution is found, if any, or until no further possibilities remain. As an implementation note, each search step gives only one additional assigned variable. Thus the quantities needed to estimate the solubility and cost proxy do not change much from step. This observation allows creating an optimized version of the heuristics that cache the required information on the subproblems. In particular, for graphs with fixed connectivity there are, on average, only a constant number of neighbors for each node. Thus making an assignment to a particular node will, on average, only affect the number of allowable assignments associated with a small number of

[4] Additional experiments based on a decreasing ordering of N_{soln} gave similar, but slightly worse, results compared to using p_n.

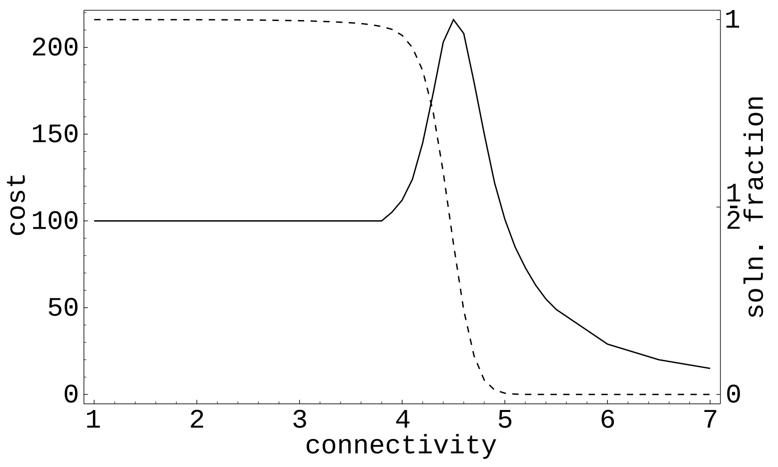

Figure 1: Behavior for 3-coloring of random graphs with 100 nodes as a function of connectivity γ in steps of 0.1, with 50,000 samples at each point. The solid curve shows the median search cost, and the dashed one is the fraction of graphs with a solution (ranging from one on the left to zero on the right).

neighbors, hence allowing rapid updates. Thus the overhead associated with this method can be made a small constant factor as larger graphs are considered. While some caching was done in the experiments reported below (based on the information already retained for the Brelaz heuristic), additional caching of information specific to the theory was not included, resulting in considerably more overhead than necessary. Running on a Sun SPARC 10, for 100-node graphs with γ between 3.5 and 4.5, the program ran about 20,000 Brelaz steps per second and 5000 steps per second when using the hardness-based value ordering.

5.1 Soluble cases

A comparison of the hardness-based value ordering heuristic with the standard Brelaz method for 3-coloring random graphs that have solutions is shown in Fig. 2 and 3. In all cases, variable ordering was that of the Brelaz heuristic. The figures show the cost distribution, i.e., the fraction of samples with cost below the indicated values. This gives a better comparison than simple aggregate measures, such as means or medians, due to the high variance in the costs. Note that in comparing distributions from two methods, the lower curve indicates better performance in that there are fewer high cost instances. For soluble problems, the minimum possible cost value is n, obtained if all value choices led to soluble subproblems so no backtrack is required. In particular, the figures show that the hardness-based heuristic produces a lower curve and increases the likelihood of backtrack-free search. Quantitatively, the χ^2 test determines the likelihood the two samples could have come from the same distribution [22]. Small values indicate the two methods are unlikely to give the same performance distribution. For $\gamma = 3.5$ this shows a definite improvement. Near the transition, at $\gamma = 4.5$ the distributions do not differ significantly.

A further example, for $\gamma = 5.5$, is shown in Fig. 4. With this many edges, most random graphs do not have solutions. So instead, these graphs were created with a prespecified solution (i.e., the nodes were divided into three sets, as close as possible to equal sizes, and no edge was permitted between nodes within the same set). This results in somewhat easier graphs, but exhibits the same transition phenomenon and at approximately the same location as random graphs [30].

To demonstrate the robustness of these results, we examined a number of other cases. These include more sophisticated searches, different problem sizes and alternate problem ensembles. For instance, Fig. 5 repeats Fig. 2 but using a sophisticated backtracking method called backjumping [7, 24]. Backjumping improves on simple backtracking by keeping track of which previous variable assignments are actually relevant for causing a conflict. Then, instead of backtracking to the most recently assigned variable (which, if not relevant for causing the conflict, simply means the search that identified the conflict will be repeated), it backtracks to the most recent relevant assignment. In the specific version used here, the search maintains a list of relevant prior assignments, which is initially empty. Then, whenever the search is forced to backtrack because there

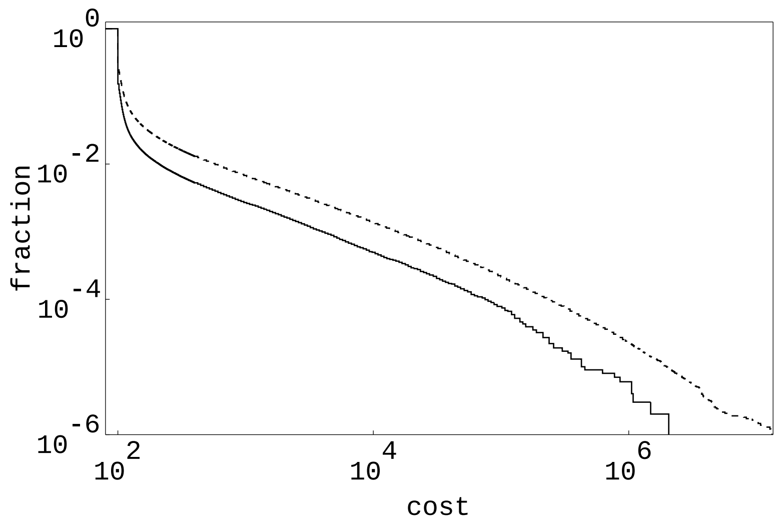

Figure 2: Distribution of search costs for random graphs that have a solution, $n = 100$ and $\gamma = 3.5$. These used the hardness-based value ordering heuristic (solid curve) and the standard Brelaz heuristic (dashed), based on 10^6 and 10^7 samples respectively. The hardness and Brelaz heuristics respectively completed 84.7% and 75.4% of the searches without backtracking (i.e., using 100 steps). The χ^2 test gives the probability of these coming from the same distribution as negligibly small, around 10^{-4800}.

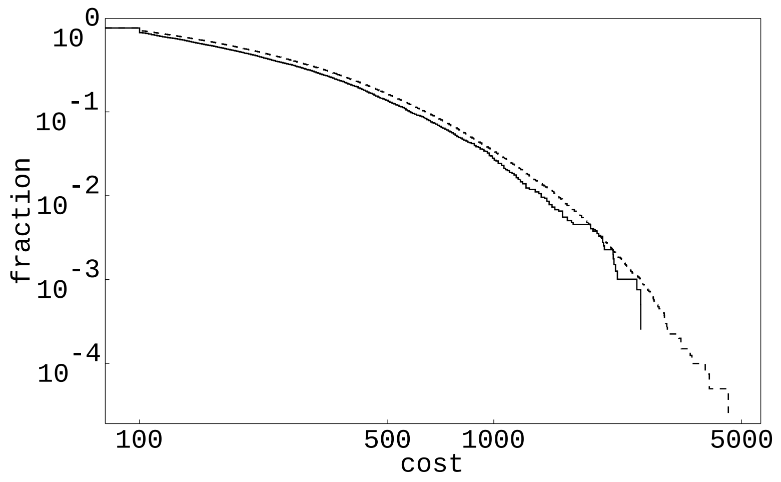

Figure 3: Distribution of search costs for random graphs that have a solution, $n = 100$ and $\gamma = 4.5$. These used the hardness-based value ordering heuristic (solid curve) and the standard Brelaz heuristic (dashed), based on 10,000 and 50,000 samples respectively. The hardness and Brelaz heuristics respectively completed 12.1% and 7.5% of the searches without backtracking (i.e., using 100 steps). The χ^2 test gives the probability of these coming from the same distribution as 40%.

are no consistent values for variable v, we add to the list any currently assigned neighbors of v (since these are the values that together cause the conflict), remove v itself from the list and then backtrack to the most recently assigned variable in the list.

Comparing the two figures shows that backjumping itself significantly improves the search, but the hardness heuristic gives an additional improvement. In fact, unlike Fig. 2, in this case the hardness heuristic also appears to increase the scaling power by which the distribution decreases for high cost cases. E.g., a fit to the distribution over the range of costs from 300 to 3000 gives a slope of -1.85 for hardness with backjumping compared to -1.45 for backjumping alone. For Fig. 2, the corresponding slopes are -0.71 and -0.70 respectively, showing no significant difference. The hardness heuristic also shows an improvement for

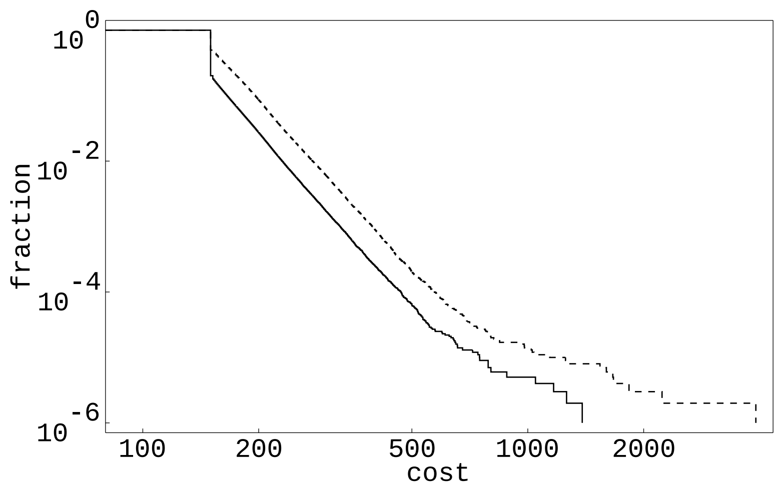

Figure 4: Distribution of search cost for graphs with a prespecified solution, $n = 150$ and $\gamma = 5.5$. These used the hardness-based value ordering heuristic (solid curve) and the standard Brelaz heuristic (dashed), based on 10^6 samples in each case. The hardness and Brelaz heuristics respectively completed 79.8% and 50.3% of the searches without backtracking (i.e., using 150 steps). The χ^2 test gives the probability of these coming from the same distribution as negligibly small, around 10^{-45000}.

the overconstrained cases with $\gamma = 5.5$. However, for $\gamma = 4.5$, the backjumping itself makes little change in performance.

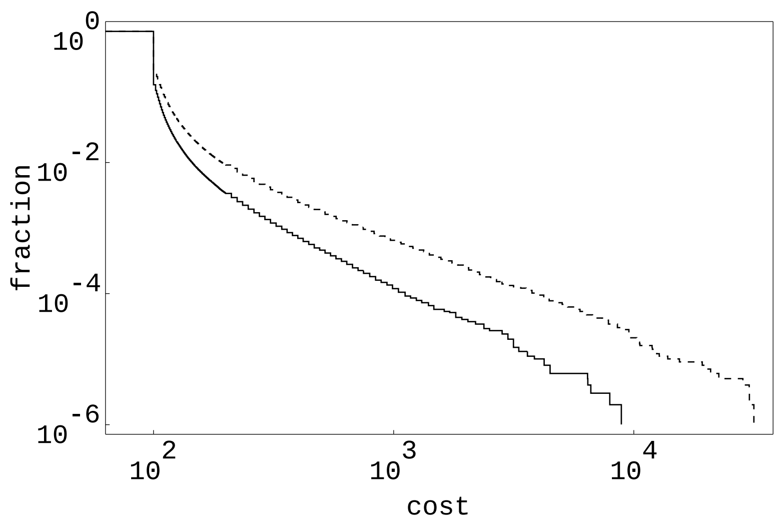

Figure 5: Distribution of search costs for random graphs that have a solution, $n = 100$ and $\gamma = 3.5$. These used the hardness-based value ordering heuristic (solid curve) and the standard Brelaz heuristic (dashed), both with backjumping method, based on 10^6 samples. The hardness and Brelaz heuristics respectively completed 84.6% and 75.5% of the searches without backtracking (i.e., using 100 steps). The backjumping cases finishes without backtracking about as often as regular case; as expected since backjumping only has a chance to operate when there is backtracking.

The behavior shown in the figures also applies to graphs with 50 nodes, indicating that the difference does not appear to depend greatly on the size of problems considered. The improvement is also seen in other cases including larger domain sizes (e.g., 5-coloring), using reduction operators [3] to eliminate trivial subgraphs before searching, and using the somewhat harder graphs in which each node is constrained to have at least b neighbors. These additional experiments suggest that the improvement is robust with respect to different ways of generating the graphs. This is particularly noteworthy in that the theory used to derive Eq. (2) assumed simple random problems. Other studies using heuristics based on estimates of problem hardness have also found robust results when applied to a variety of search problems [9].

In summary, the value-ordering hardness-based heuristic clearly helps for under- or overconstrained examples, but is less helpful for the transition cases. To investigate this point further, we examined the pairwise difference in the two search methods at $\gamma = 4.5$, further reducing the variance between them by having both methods use the same variable ordering choices when more than one variable was rated equally good by the Brelaz heuristic. Specifically, 10,000 graphs were generated, and the cost of both search methods was recorded for each graph. This provides a much more sensitive discrimination between the two methods than just the distributions shown in Fig. 3 which include variation in individual problems as well as between the two methods. Quantitatively, the paired t-test [22] on this data gives a probability of 10^{-7} that the two methods produce the same mean search cost. Combined with the slightly lowered mean cost when using the hardness based heuristic (299 steps instead of 326 for the Brelaz heuristic), this shows that the improvement, while smaller than for $\gamma = 3.5$ or $\gamma = 5.5$, is still statistically significant.

5.2 Insoluble cases

For insoluble cases, value ordering is not useful: all values will need to be examined during the search. We thus consider the use of the cost proxy for variable ordering, specifically to break any ties in the variable ordering suggested by the Brelaz heuristic. Comparing the distributions for 50 and 100-node graphs with no solutions shows no statistically significant difference for γ equal to 3.5, 4.5 and 5.5. Nevertheless, comparing the two search methods on the same set of 100-node graphs for γ of 4.5 and 5.5 shows that use of the cost proxy to order the variables results in about a 5% reduction in the mean search cost compared to the Brelaz heuristic alone. Although this improvement is relatively small, it is statistically significant in that the paired t-test rejects the hypothesis that the two search methods are the same at a significance level of about 10^{-20}. For $\gamma = 3.5$ almost all graphs are soluble resulting in too few insoluble samples to make a statistically significant comparison.

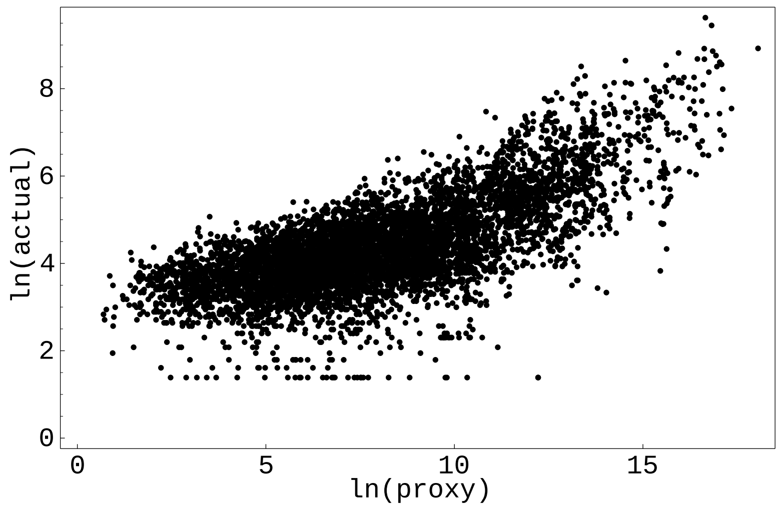

Figure 6: Actual values of $\ln(C_{1st})$ vs. the estimate from the proxy described in §2.2 for the 6380 subproblems obtained during 100 searches of 200-node random graphs with $\gamma = 5.0$. The actual search cost generally increases as the proxy does, but with considerable variance.

One possible reason for the limited improvement is the relatively conservative use of the cost information, i.e., it is only used when the Brelaz heuristic rates more than one variable as equally good next choices. This results in relatively few opportunities to actually use the cost proxy, specifically in about 4% of the variable choices for the 100–node graph searches. Nevertheless, limited experiments with exhaustive search to determine the actual best variable choice shows that there is indeed room for considerable reduction in search cost from simply making better choices in the few cases where the Brelaz heuristic results in ties.

Using the cost proxy achieves only a small portion of this potential. This is due to the large variance in the actual costs compared to the proxy value, as shown in Fig. 6.

In a final set of experiments, we examined a less conservative use of the cost proxy to order the variable choices. Specifically, instead of just breaking any ties among the top rated nodes of the Brelaz heuristic, we considered all nodes with the fewest remaining available colors and used the cost proxy to select the next one to try. This expands the set of nodes considered with the cost proxy (to about 8% of the variable choices for 100-node graphs) since the Brelaz heuristic selects, from among those nodes with the fewest remaining choices, those with the most uncolored neighbors. For 100-node insoluble graphs with γ equal to 4.5 and 5.5, this gave slightly worse average performance than the standard Brelaz method. The paired t-test in these cases showed that this decrease in performance was statistically significant with a probability of less than 10^{-14} that the search methods have the same average behavior. This further supports the observation that the cost proxy is not sufficiently precise to discriminate well among variable choices. Instead, local information should be used as much as possible, and the global cost proxy used only when local information results in tied choices.

6 Discussion

In summary, solubility estimates improve the value ordering choices, although most usefully for problems away from the transition region. By contrast, the cost proxy estimate is not sufficiently precise to more than slightly improve the variable ordering choices. This suggests that additional parameters for the problem structure, such as the amount of constraint clustering [13], are required to reduce the variance in the cost estimates. Interestingly, this variance, particularly large near the transition region for these problems, provides a computational consequence of the large fluctuations often associated with physical phase transitions.

A number of more general questions remain. First, since the theory readily applies to any CSP, it is of interest to evaluate its usefulness for other CSPs such as satisfiability (SAT) or random binary CSPs whose transition behaviors have been extensively studied. Second is the question of whether this behavior can be exploited with more complex backtrack methods [6, 10, 24], beyond the addition of backjumping, or problem decomposition [8], perhaps especially to reduce difficulties that these methods can introduce [1, 23]. Finally, this work provides an example of how the underlying nature of problems can be used to devise better search methods. This thus complements other studies using the degree of constraint as a heuristic [9] or exploiting transition behaviors in genetic algorithms [4], optimization problems [21] and machine vision [11].

More generally, this body of work suggests a broad new application for simple "mean-field" theories in computation. That is, even though the particular subproblems encountered during search depend specifically on the previous choices made, treating these problems as if they are independently and randomly generated is accurate enough to give observable improvements in performance.

References

[1] Andrew B. Baker. The hazards of fancy backtracking. In *Proc. of the 12th Natl. Conf. on Artificial Intelligence (AAAI94)*, pages 288–293, Menlo Park, CA, 1994. AAAI Press.

[2] C. Berge. *Graphs and Hypergraphs.* North-Holland, Amsterdam, 1973.

[3] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the really hard problems are. In J. Mylopoulos and R. Reiter, editors, *Proceedings of IJCAI91*, pages 331–337, San Mateo, CA, 1991. Morgan Kaufmann.

[4] Scott H. Clearwater and Tad Hogg. Problem structure heuristics and scaling behavior for genetic algorithms. *Artificial Intelligence*, 81:327–347, 1996.

[5] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in random 3SAT. *Artificial Intelligence*, 81:31–57, 1996.

[6] R. Dechter. Learning while searching in constraint-satisfaction problems. In *Proc. of AAAI-86 5th Natl. Conf. on AI, 11-15 Aug 1986*, pages 178–183, Los Altos, CA, 1986. Morgan Kaufmann.

[7] Rina Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990.

[8] Eugene C. Freuder and Paul D. Hubbe. Extracting constraint satisfaction subproblems. In *Proc. of ECAI94*, 1994.

[9] Ian P. Gent, Ewan MacIntyre, Patrick Prosser, and Toby Walsh. The constrainedness of search. In *Proc. of the 13th Natl. Conf. on Artificial Intelligence (AAAI96)*, pages 246–252, Menlo Park, CA, 1996. AAAI Press.

[10] Matthew L. Ginsberg. Dynamic backtracking. In Haym Hirsh et al., editors, *AAAI Spring Symposium on AI and NP-Hard Problems*, pages 64–70. AAAI, 1993.

[11] Craig Gotsman. A cluster detection algorithm based on percolation theory. *Pattern Recognition Letters*, 12:199–202, April 1991.

[12] Tad Hogg. Phase transitions in constraint satisfaction search. A World Wide Web page with URL http://www.parc.xerox.com/dynamics/www/constraints.html, 1994.

[13] Tad Hogg. Statistical mechanics of combinatorial search. In *Proc. of the Workshop on Physics and Computation (PhysComp94)*, pages 196–202, Los Alamitos, CA, 1994. IEEE Press.

[14] Tad Hogg, Bernardo A. Huberman, and Colin Williams. Phase transitions and the search problem. *Artificial Intelligence*, 81:1–15, 1996.

[15] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning. *Operations Research*, 39(3):378–406, May-June 1991.

[16] Scott Kirkpatrick and Bart Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264:1297–1301, 1994.

[17] Vipin Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992.

[18] Alan Mackworth. Constraint satisfaction. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 285–293. Wiley, NY, 1992.

[19] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[20] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In *Proc. of the 10th Natl. Conf. on Artificial Intelligence (AAAI92)*, pages 459–465, Menlo Park, 1992. AAAI Press.

[21] Joseph C. Pemberton and Weixiong Zhang. Epsilon-transformation: Exploiting phase transitions to solve combinatorial optimization problems. *Artificial Intelligence*, 81:297–325, 1996.

[22] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes.* Cambridge Univ. Press, Cambridge, 1986.

[23] Patrick Prosser. Domain filtering can degrade intelligent backtracking search. In *Proc. of IJCAI93*, pages 262–267, San Mateo, CA, 1993. AAAI, Morgan Kaufmann.

[24] Patrick Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9(3):268–299, 1993.

[25] Patrick Prosser. An empirical study of phase transitions in binary constraint satisfaction problems. *Artificial Intelligence*, 81:81–109, 1996.

[26] Muhammad Sahimi. Progress in percolation theory and its applications. In D. Stauffer, editor, *Annual Reviews of Computational Physics, vol. 2*, pages 175–241. World Scientific, Singapore, 1995.

[27] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proc. of the 10th Natl. Conf. on Artificial Intelligence (AAAI92)*, pages 440–446, Menlo Park, CA, 1992. AAAI Press.

[28] Barbara M. Smith and Martin E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81:155–181, 1996.

[29] Colin P. Williams and Tad Hogg. Extending deep structure. In *Proc. of the 11th Natl. Conf. on Artificial Intelligence (AAAI93)*, pages 152–157, Menlo Park, CA, 1993. AAAI Press.

[30] Colin P. Williams and Tad Hogg. Exploiting the deep structure of constraint problems. *Artificial Intelligence*, 70:73–117, 1994.

[31] D. C. Wood. A technique for coloring a graph applicable to large-scale optimization problems. *Computer Journal*, 12:317, 1969.