Where the *Really* Hard Problems Are

Peter Cheeseman
RIACS*

Bob Kanefsky
Sterling Software

William M. Taylor
Sterling Software

Artificial Intelligence Research Branch
NASA Ames Research Center, Mail Stop 244-17
Moffett Field, CA 94035, USA

Email: < last-namO@ptolemy.arcnasa.gov

Abstract

It *is* well known that for many NP-complete problems, such as K-Sat, etc., typical cases are easy to solve; so that computationally hard cases must be rare (assuming P = NP). This paper shows that NP-complete problems can be summarized by at least one "order parameter", and that the hard problems occur at a critical value of such a parameter. This critical value separates two regions of characteristically different properties. For example, for K-colorability, the critical value separates overconstrained from underconstrained random graphs, and it marks the value at which the probability of a solution changes abruptly from near 0 to near 1. It is the high density of well-separated almost solutions (local minima) at this boundary that cause search algorithms to "thrash". This boundary is a type *of* phase transition and we show that it is preserved under mappings between problems. We show that for some P problems either there is no phase transition or *it* occurs for bounded N (and so bounds the cost). These results suggest a way of deciding if a problem is in P or NP and why they are different.

1 Introduction

A common result *of* AI research is to show that *some* class of problems is NP-complete (or NP-hard), with the implication that this class of problems is very hard to solve (assuming P = NP). On the other hand it is well known that for many of these *NP* problems, typical instances are easy to solve (e.g. [10]). There is no contradiction here, since NP complexity is usually a worst case analysis for a whole class of problems, and so says nothing about the difficulty of typical instances- However, this situation raises the question "where are the really hard instances of NP problems?" Can a subclass of problems be defined that is typically (exponentially) hard to solve, or do worst cases appear as rare "pathological cases" scattered unpredictably in the problem space?

*Research Institute for Advanced Computer Science

In this paper we show that for many NP problems one or more "order parameters" can be defined, and hard instances occur around particular critical values of these order parameters. In addition, such critical values form a boundary that separates the space of problems into two regions. One region is underconstrained, so the density of solutions is high, thus making it relatively easy to find a solution. The other region is overconstrained and very unlikely to contain a solution. If there are solutions in this overconstrained region, then they have such deep local minimum (strong basin of attraction) that any reasonable algorithm is likely to find it. If there is no solution, then a backtrack search can usually establish this with ease, since potential solution paths are usually cut off early in the search. Really hard problems occur on the boundary between these *two* regions, where the probability of a solution is low but non-negligible. At this point there are typically many local minima corresponding to almost solutions separated by high "energy barriers". These almost solutions form deep local minima that may often trap search methods that rely on local information.

Because it is possible to locate a region where hard problems occur, it is possible to predict whether a particular problem is likely to be easy to solve. We expect that in future computer scientists will produce "phase diagrams" for particular problem domains to aid in hard problem identification and for prediction of solution existence probability, such as shown in [6].

We present these ideas by first showing how phase transitions arise in problem solving, and then illustrating particular transitions through several examples with different properties. We then show how some of these examples interrelate when they are mapped onto each other. Finally, we summarize the results and state a strong conjecture based on these results.

2 Phase Transitions

We first review *well-studied cases where the* behavior of a complex system, including phase transitions, can be described by an order parameter. For example, the probability that a random graph is connected, or contains a Hamilton circuit, or a triangle etc., has a sharp threshold for particular values of the average graph connectivity. In the case of graph connectivity and Hamilton circuits, this threshold depends on the graph size as well. Other

properties of random graphs also show interesting behavior around the transition point which is characteristic of phase transitions. In particular, the size of the largest connected sub-graph grows very rapidly as a function of the average connectivity as the critical connectivity is approached from below. Also, the sizes of the subgraphs below the threshold show a fractal distribution—these properties are related to analogous physical systems, e.g.

Our interest in phase transitions arises from the discovery that hard to solve problems occur at such boundaries for many types of problems. The importance of phase transitions for AI is discussed in [4] where it is argued that complex systems composed of many interacting values can often be understood at the macroscopic level in terms of a few order parameters that are characteristic of the system as a whole. Summarizing the properties of complex systems through a small set of parameters is routine in statistical mechanics [2], [7]. This is possible because a large number of local interactions can produce dramatic coordinated macroscopic behavior, such as phase transitions, that do not depend on the detailed interactions within the system. Examples of phase transitions in AI are given in [5],[6].

3 An Example: Hamilton Circuits

A Hamilton Circuit (HC) is a cyclic ordering of a set of nodes such that there is an edge connecting every pair of nodes in the graph in order. The cyclic condition ensures that the circuit is closed, and the requirement that all the nodes be included (with no repeats) ensures that the circuit does not cross over itself, and passes through every node. The problem is to find if a HC exists for a given graph.

The first question we investigate is how the probability of the existence of a HC in a random graph varies with the average connectivity of the graph. The results for several different graph sizes are shown in Fig. Ia. These results were generated by finding the proportion of 20 randomly generated graphs that contained a HC for graphs with different connectivities and numbers of nodes. A fully connected graph always has a HC (all node orderings are a HC), and so an almost fully connected graph has a very high probability of containing a HC. In this region there are a very large number of HCs, and this number drops rapidly as the boundary is approached. At the other extreme, a random graph barely above an average connectivity of 2 is unlikely to even be connected, and so is very unlikely to contain a HC. For some critical value of the average connectivity between these two extremes, the probability of a HC changes steeply from almost 0 to almost 1. Theory predicts that the transition will occur at an average connectivity of $\ln N + \ln \ln N$ [1], and this prediction is supported by our results in Fig. Ia, where the theoretical value is shown by the dashed vertical line. This transition shows the characteristic properties of a phase transition. For example, the size of the largest almost HC grows exponentially below the threshold, and this is the main reason for our next result. The second question we investigate is how the computational cost of

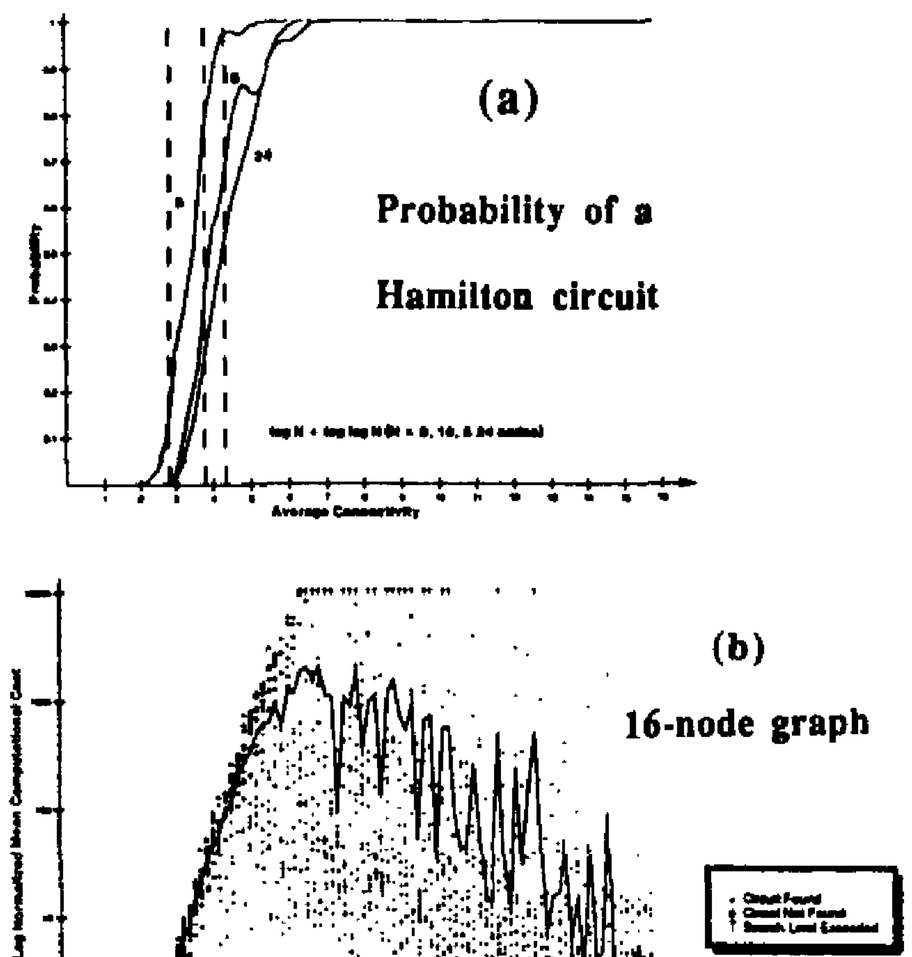

Figure 1: Solution Probability and Cost for Ham. Cir.

finding a HC (if one exists) varies with the connectivity. Graphs were randomly generated with a given connectivity, and a backtrack search procedure was run for each graph. The number of steps this procedure needed to find a HC relative to the minimum number is shown in Fig. Ib,c. The arrows at the top of the figure represent graphs whose cost exceeded a prespecified maximum. The solid line represents the average of 20 trials, but since this average includes the saturated values it severely underestimates the true cost. Despite this underestimation, the existence of a phase transition is clear. The sharpness of the transition increases with increasing graph size (N), as does the location of the critical connectivity. Similar results were obtained for graphs that were guaranteed to contain a HC by construction. An important effect to note in both cases is that the phase transition in cost occurs at the same point at which the probability of a HC drops to zero within the numeric

accuracy.

The HC backtrack algorithm uses two heuristics: 1) the initial starting node is the one with the highest connectivity, 2) neighbor nodes of the last node selected are sorted by their connectivity to the remaining unselected nodes—the neighbor with the highest connectivity is selected first. More effective heuristics, such as bidirectional search, may exist, but we do not expect that the behavior observed in Fig. Ib,c would be qualitatively different.

For HCs, the above results show that the average connectivity is an appropriate order parameter, but is this the only one? Another possible order parameter is the *variance* of the average connectivity. Preliminary results show that the phase transition exists for both high and low variance_t but the data is insufficient to decide if the variance changes the location of the critical connectivity.

We have shown empirically that for HCs there is a phase transition in computational cost around a critical value of the average connectivity of a graph. The value of this critical connectivity increases with graph size (N), and also gets sharper with N. This critical connectivity occurs at the point at which the probability that there is a HC in a random graph drops to almost zero. In other words, the critical connectivity separates two regions. 1) A low connectivity region where there is almost zero probability of there being a HC—here the backtrack algorithm quickly terminates because all potential HCs are cut off early in the search. If there is a solution in this low probability region, then the algorithm has little trouble finding it, since every other alternative is cut off early. 2) A high connectivity region, on the other hand, has a high density of HCs, and so the backtrack algorithm quickly finds one. It is on the border between these two regions where hard problems occur. On the border there are many almost HCs that are quite different from each other (i.e. it typically takes many changes to transform an almost solution into any other almost solution) and these numerous local minima make it hard to find a HC (if there is one). Any search procedure based on local information will have the same difficulty.

Although the most difficult HC problems occur in the neighborhood of the phase transition, this does not mean that problems not near the transition are easy. For example, strictly 3-connected random graphs with at least one HC (guaranteed by construction) have a solution time that grows exponentially with the size of the graph (using the above backtrack algorithm). This means that even HC problems on random graphs whose connectivity is some distance from the phase transition are very hard—but those on the boundary are even harder. The reason most HC problems are hard, whereas most graph coloring problems are not *seems to be* a consequence of the global constraint in HCs, but not for K-colorability.

For HC problems there is one global constraint (i.e. a constraint involving all the nodes)—perhaps the phase transition is a result of this property? To investigate this possibility, we next examine graph coloring as an example with only pair-wise local constraints.

4 An Example; Graph Coloring

This is a constraint satisfaction problem, where each variable can take on a number of possible values ("colors"*),* and there are binary constraints that forbid particular pairs of variables from having the same color. The goal is to see if there is an assignment of colors to the variables that satisfy the constraints and only use *K* colors, or report that no assignment is possible. This version is the *K* colorability decision problem; more generally the goal is to find the minimum *K* that satisfies the constraints (the chromatic number problem). Any solution to a graph coloring problem can be used to generate other solutions by interchanging the colors, implying a color rotation symetry. Many practical constraint satisfaction problems, such as timetable construction, can be mapped into a graph coloring problem.

Graph coloring has been extensively investigated, both theoretically and empirically, e.g., [9],[10]. Even though graph coloring is an NP-complete problem, these authors report that graph coloring is "almost always easy". In particular, a simple backtrack algorithm by Brelaz was found to solve all randomly generated graphs it was tried on with little backtrack [10]. We continue these investigations but restrict our attention to random graphs that have been "reduced". The ^reduction operators" decribed below guarantee that if the reduced graph is It-colorable (or not) then the original graph is K-colorable (or not). Any graph that can be reduced to one that is trivially K-colorable (or not) can be solved without search. We only investigate the space of reduced graphs, because the hard problems must be in this space. The particular reduction operators we used for K-colorabi!ity are:

1. Remove Underconstrained Nodes—a node with less than *K* constraints can be removed, because it can always be colored.
2. Remove Subsumed Nodes—a node N can be removed if there is a node M that is connected to everything N is connected to, since any color that works for M will work for N (provided N is not connected to M).
3. Merge nodes that must have the same color—if any nodes are fully connected to a clique of size K-I, then these nodes can be merged into a single node with all the constraints of its constituents, because they must have the same color.

Underconstrained: Redundant Symmetric:

Figure 2: Reduction Operators for 3-Colorability

These reduction operators can be applied in any order, and typically the application of one operator creates a

situation where other operators become Applicable, producing a reduction cascade. We found that these operators reduced all our carefully hand-constructed "hard" graphs to trivial cases! In particular, all graphs reduced using *K = 1* reduced to the null graph, showing that 2-colorability is a trivial P problem [3], Although these are all the reduction operators we could find, they are all that is possible. In particular, we have found reduction operators that eliminate more than one node at a time, but these operators are so rarely applicable we did not use them. The following investigations are all in the space of random reduced graphs because the K-colorability of a reduced graph is equivalent to that of many unreduced graphs. This kind of problem simplification by preprocessing is often overlooked in discussion of algorithms, yet it can make apparently hard problems trivial. The essential difference between problem reduction and problem solving is that problem reduction does not produce disjunctive alternatives (i.e. no search).

We empirically investigated the probability of a solution for K-colorability problems for different values of K and N (number of nodes). The results are shown in Fig. 3a, where each probability point is the average of about 5 trials, but there are no points in the transition region because they are too costly to compute. Two trends are clear from these results. First is the abrupt change in solution probability occurs at higher values of the connectivity for larger K, and the other is the sharpness of the transition increases with N.

We next show how the computational cost varies as a function of the connectivity for different values of K. The results are shown in Figs. 2b,c for random reduced graphs that were generated so that they were guaranteed to have a solution. For both 3-Col and 4-Col the existence of a phase transition is clear, and their location is the same as that for the corresponding solution-probability transition to within the numeric noise level. The transition for.4-Col is much sharper than for 3-Col. Similar results are obtained for random reduced graphs that are not guaranteed to have a K-col solution, and for random graphs restricted to 2-D neighbor connections.

Brelaz's algorithm [10] uses heuristics: Select an uncolored node with the fewest remaining colors; ties are broken by selecting the node connected to the most uncolored nodes; remaining choices are made randomly. This is a very effective algorithm, but its performance at the phase boundary is highly variable, even on the *same* graph, because the heuristics do not always lead to a unique choice. These "fluctuations" are typical of behavior near a phase transition. This observation suggests an improvement for a backtrack algorithm—run many versions of it in parallel, so that the expected number of steps is lower than for a single version.

These results show that there is a phase transition for the cost of solving K-colorability problems, and it occurs at the critical average connectivity where the probability of a solution has dropped to almost zero. This explains why previous authors [9],[10] found K-colorability an easy problem—they were using *nonreduced* graphs whose effective connectivities did not typically fall near the phase boundary.

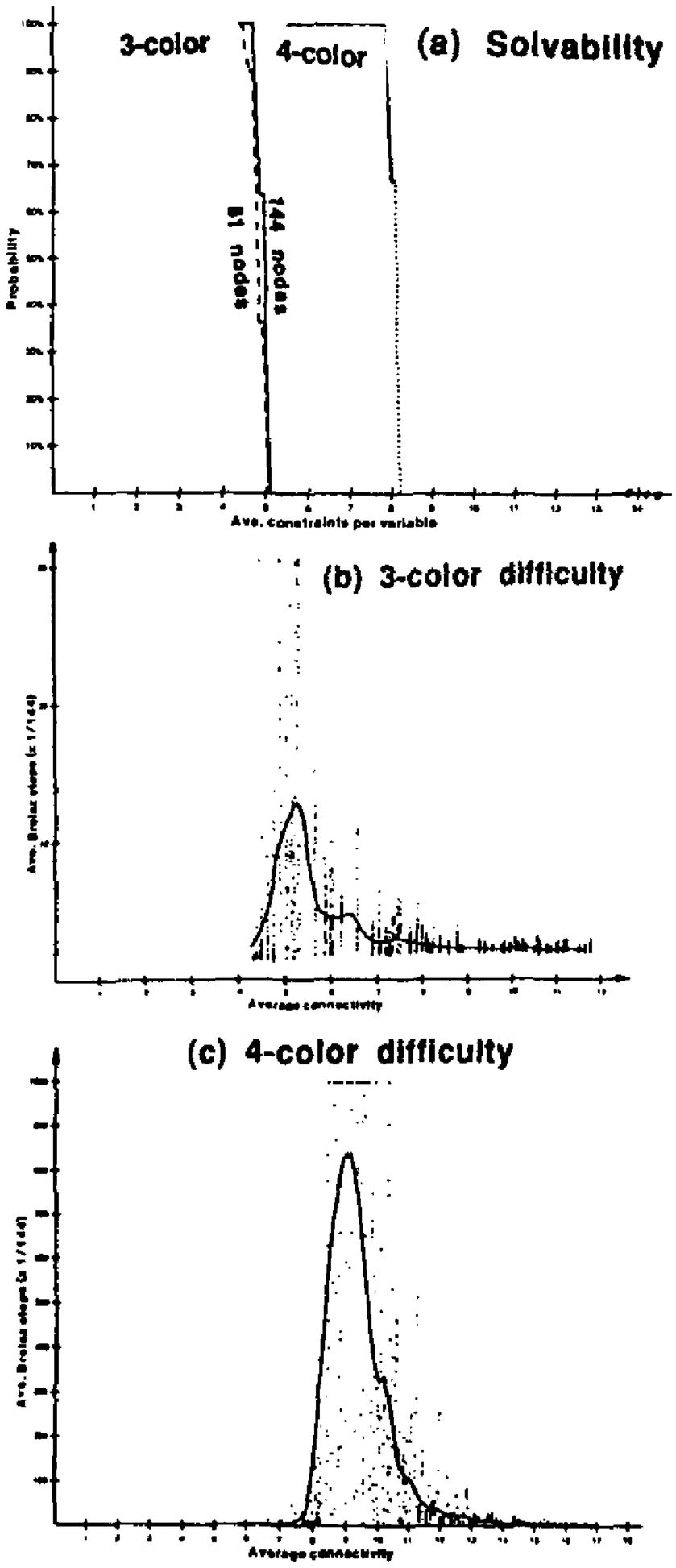

Figure 3: Solution Probability and Cost for K-Col

Examination of Brelaz's algorithm on the hard instances shows that it often backtracks, and sometimes backtracks all the way to the beginning. This "thrashing* occurs because there ate many near-solutions available, and they look like solutions to the backtrack algorithm until an assignment is nearly complete. These local minima make it hard to find a solution if one is present—the proverbial needle in a haystack. This observation hints as to why some NP instances are hard; it seems likely that any algorithm based on local informa-

tion will be fooled by the high number of local minima in problems near the phase boundary. Further experimentation suggests that graphs with a high variance of the average connectivity are generally easier to solve than ones with lower variance, so that the variance of the average connectivity may be an additional order parameter.

An apparent exception to the conjecture that all the NP problems overlap the critical boundary, is to discover if planar graphs with node connectivity < 4) are 3-CoJ. This is an NP-complete problem [3]$_>$ yet our 3-Col transition for random reduced graphs occurs at a connectivity ~ 5.4 (see Fig. 3b)), To compare planar graphs with our results we must first reduce them. Merging destroys the planarity of the graph, reduces the number of nodes and simultaneously increases the connectivity of the merged nodes. The results is increased average connectivity of the previously planar graphs so that they straddle the critical connectivity, thus preserving the conjecture. However, if the proportion of 4-connected nodes in 3,4 planar graphs are sufficiently restricted, the result is a new P class of problems, assuming the conjecture is true.

5 An Example: K-Satisfyability

Since it is possible to map K-colorability problems into K-sat problems and vice versa [3], we next investigate K-sat problems. We used a form of resolution to reduce random K-sat problems before applying a simple backtrack search procedure with a most-constrained-first heuristic. The results are shown in Fig. 4. As for k-colorability, there is a sharp drop in the probability of a solution at some critical value of the graph average connectivity, and this critical value depends on *K.* Also, the normalized cost of solution shows a phase transition at about the point at which the solution probability drops to near zero. Sirmlar results were found for random reduced graphs whose method of construction guaranteed at least one solution. We found that 2-sat does not have a phase boundary, but 3-aat does, as expected, since 2-sat is P and 3-sat is NP.

Large random reduced K-sat problems seem to get easier with large K—the phase transition evident in Fig. *4* becomes weaker with larger K until it disappears altogether (i.e. the backtrack search procedure solves the problem without backtrack). We do not fully understand this behavior, but it is probably due to an incomplete set of reduction operators that leave so may "trivial" problems (for large *K)* that the relatively few hard problems are missed. This is the same reason that previous authors missed the phase transition for K-colorability; the hard problems are greatly diluted by trivial problems. So far, the only way we have been able to create hard K-sat problems is by mapping equivalent K-colorability problems.

6 An Example: Traveling Salesman

All the previous examples have been constraint satisfaction problems where the order parameter turned out to

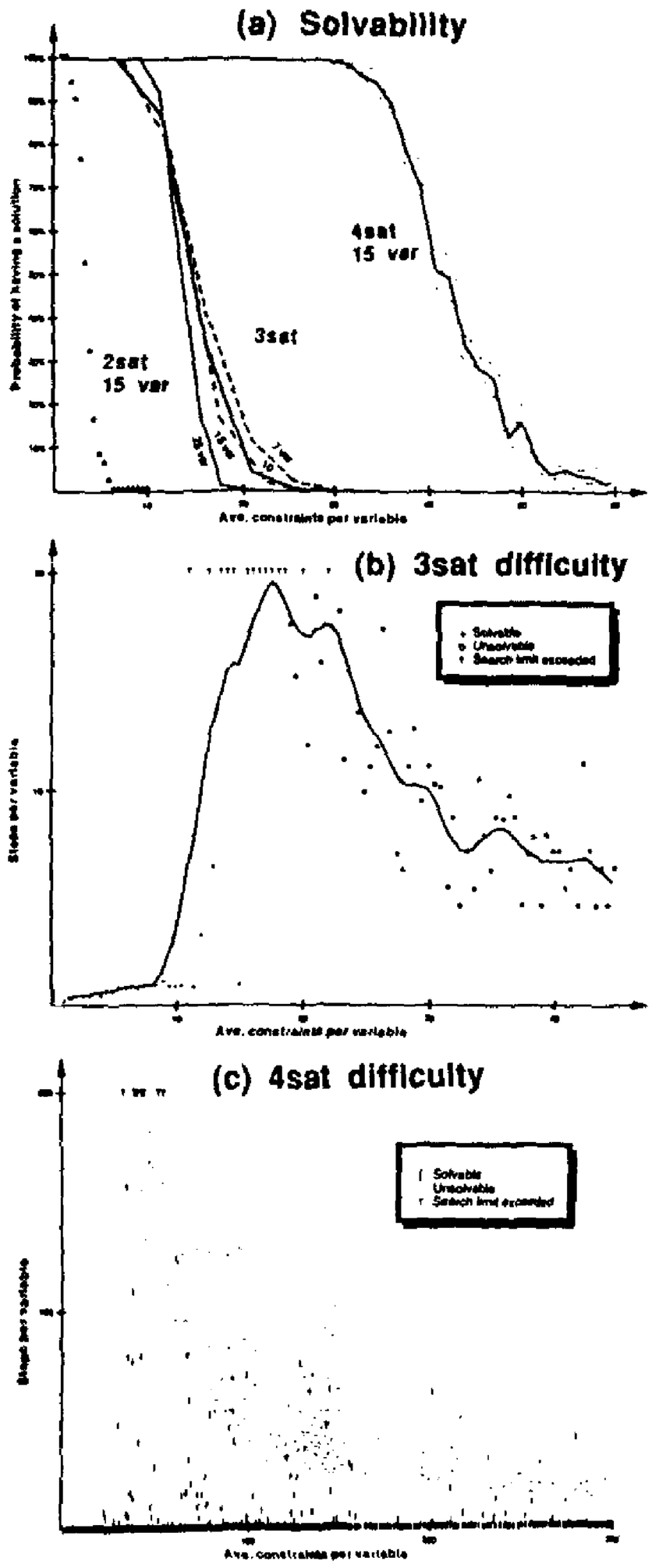

Figure 4: Solution Probability and Cost for K-sat

be the average connectivity of the corresponding graph, and the variance of the average connectivity may be a weak additional order parameter. We now investigate a different order parameter (the standard deviation of the cost matrix) in the context of a minimization problem—the Traveling Salesman Problem (TSP). In a TSP, the goal is to find a Hamilton circuit among a set of nodes ("cities") such that the total cost of the circuit is a minimum. The costs of edges in the graph are given by an

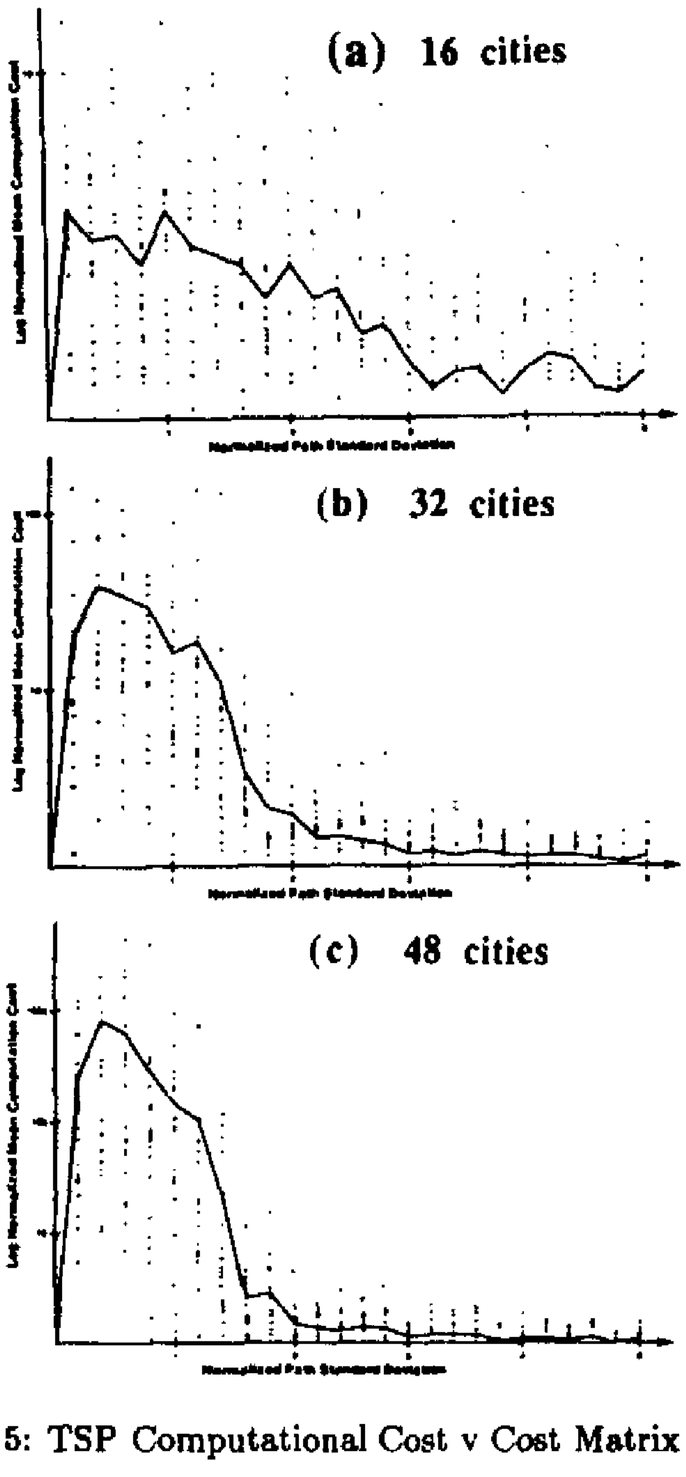

Figure 5: TSP Computational Cost v Cost Matrix S.D.

interger-valued cost matrix that in general is not symmetric. This cost matrix can be rescaled and a constant added without changing the essential problem. For convenience we choose cost matrices with a mean edge cost of 10, but with varying standard deviations of these costs. To estimate the computational cost of solving TSP problems we used Little's algorithm; the best exact algorithm we could find [8], It is a kind of backtrack algorithm that efficiently exploits properties of the cost matrix and guarantees to find a minimum cost solution.

The results of running Little's algorithm for different numbers of cities with random cost matrices constructed according to a log-normal distribution with the given standard deviation are shown in Figs. 5 a,b,c. Note that the vertical axis is a Log scale, so the phase transition is more dramatic than appears at first sight. The magnitude and sharpness of the phase transition clearly increases with city size.

Because TSP is a minimization problem, there is no probability-of-solution phase transition, such as for Hamilton circuits and graph colorability. However, as for the previous problems, the obvious phase boundary separates two distinct regions. In one region where the standard deviation is high, only the low cost tail of the distribution is considered by Little's algorithm, since any minimum cost circuit will only use these low cost transitions. At the other extreme, where the costs are mostly equal, there are many tours of the same minimum cost, and so finding one of them is not difficult. This is why the curves in Fig. 5 drop precipitously when the standard deviation goes to zero. Once again, a phase boundary has been found between two regions of fundamentally different behavior. Around this boundary the density of local minima of almost equal cost tours forces the search algorithm to investigate many false leads, leading to a dramatic rise in the computational cost.

Note that TSP contains HC as a special case where all the costs are either 1 or 2,(see [3]), where a cost of 1 corresponds to the nodes being connected. Because TSP "contains" HC as a special case, the phase transition in HC would be expected to occur at the same variance value as TSP (when the 1,2 costs are suitably mapped). This behavior has been observed to within the numerical noise limits.

7 Mappings Between Problems

Perhaps the main contribution of the NP-completeness theory is the demonstration that many apparently different problems can be mapped into each other so that solutions are preserved under the mapping. The main conjecture of this paper is that problems whose order parameter is at the critical boundary are typically really hard. If this is true, then an important question is whether the critical boundaries are preserved under these problem mappings, as would be expected if this conjecture is true.

We first investigate what happens when hard-to-K-color reduced graphs are mapped into equivalent K-sat problems. By introducing Boolean variables to represent propositions such as "Node 10 is Red", and translating the constraints into conjunctive normal form, K-col can be mapped into a K-sat problem. This direct mapping can be further reduced by applying resolution to the clauses. This shows that mapping a reduced problem in one space into an equivalent problem in another space does not necessarily produce a reduced problem in the new space. Using this mapping, we found that hard to color K-col graphs translate into hard to solve K-sat problems, as expected.

Another interesting question is what happens to a P problem if it is mapped into an NP problem in the same family? Do P problems avoid the critical region in such a mapping? As an example, consider the mapping of 2-sat (a P problem) into 3-sat (an NP problem). Such a mapping is given in [3], where a 2-sat clause, such as *(a* V b) goes into two clauses (a V *b* V *x)* and *(a* V *b* V *x).* Since every such transformation introduces an extra variable (e.g. *x)* which only occurs in two clauses, the average connectivity of all the variables is dragged below the critical threshold. In other words, just transforming 2-sat into 3-sat by trivial variable addition does not produce hard problems since the transformed problems do not overlap the critical region.

In view of these results it is tempting to conjecture that the difference between P and an NP problems is

whether a phase boundary exists or not. Unfortunately, this is not true—what matters is whether the phase boundary (if there is one) occurs at a fixed *N* or not. To explain this distinction, we compare the above results with the N-Queens problem [9], which is a known to be P. For Hamilton circuits, the phase boundary occurs at an average connectivity given approximately by ln N + lnln N, while for K-col the boundary occurs at particular values of the average connectivity that depend on *K* but not on *N.* However, for the N-Queens problem the number of variables increases as N^2, while number of constraints only increases as *O(N).* This means the average connectivity of the variables decreases with increasing *N* and cannot be freely chosen. For low *N* the problem is overconstrained and for high *N* the problem is very underconstrained. There is a phase transition at *N* = 4, but because this phase transition occurs for a fixed (low) *N,* the amount of computation is strongly bounded, as expected for a P problem,

8 Discussion

Because of the basic equivalence of NP-complete problems, we expect phase boundaries in NP complete problems other than those investigated here. However, constrained minimization problems such as graph partition, integer partition, maximal clique, Ramsey numbers etc., may have a different order parameters. Some of these problems show a "spin-glass like" transition [2].

An objection to the above results may be that they are all based on heuristic backtrack search, so that they may be a result of this choice rather than intrinsic to the problem. For graph coloring, a "local repair" algorithm [9] and a probablistic search procedure also had difficulties with reduced graphs in the critical connectivity range, adding confirmation that the phase phenomenon is intrinsic. The difficulties experienced by all these algorithms seems to be due to the large number of local minima.

9 Conclusions and Conjectures

The results reported above suggest the following conjecture:

> All NP-complete problems have at least one order parameter and the hard to solve problems are around a critical value of this order parameter. This critical value (a phase transition) separates one region from another, such as overconstrained and underconstrained regions of the problem space. In such cases, the phase transition occurs at the point where the solution probability changes abruptly from almost zero to almost 1.

The converse conjecture is:

> P problems do not contain a phase transition or if they do it occurs for bounded *N* (and so has bounded cost).

We have presented empirical evidence for these conjectures for particular problem classes, and have shown in some cases that the hard problems in one space map into hard problems in the other space, thus preserving the phase boundary under the mapping. We have also shown cases where the distinction between P and NP was whether the the P class was excluded from the critical region. If these conjectures are true, then all that is needed to turn an NP problem into a P problem is to add restrictions so problems near the critical value of the order parameter are excluded. Note that these results depend on using only reduced problems, which may explain why these particular results have not been previously noticed.

There are many outstanding questions, such as: "What happens for NP-hard problems"; "Can hard problems occur in the non-critical region?", "Do other types of problems, such as optimization problems, games, etc. have the same properties?"; and so on.

Acknowledgements

We wish to gratefully acknowledge the many stimulating discussions and ideas of W. Buntine, E. Gamble, S. Minton, A. Philips, A. Meyer, O. Hansen and B. Pell.

References

[1] Bollobas, B. *"Random Graphs",* Academic Press, London,1985.

[2] Fu, Y. *"The Uses and Abuses of Statistical Mechanics in Computational Complexity"*, in " Lectures in the Sciences of Complexity", Ed, D, L. Stein, pp 815-826, Addison Wesley, 1989.

[3] Garey M. R, and Johnson D. S., *"Computers and Intractability': A Guide to the Theory of NP-Completeness"*, Freeman, 1979.

[4] Huberman, B. A. and Hogg, T., *"'Phase Transitions in Artificial Intelligence Systems",* Artificial Intelligence, 33, 155-171, 1987.

[5] Karp, R. M. and Pearl, J., *"Searching for an Optimal Path in a Tree with Random Costs"[9],* Artificial Intelligence, (1,2), 99-116, 1983.

[6] Purdom, P. W., *"Search Rearrangement Backtracking and Polynomial Average Time"*, Artificial Intelligence, 21(1,2), 117-134, 1983.

[7] Kirkpatrick, S. and Swendsen, R. H., "Statistical Mechanics and Disordered Systems", Comm. ACM, 28, 4, 363-373, April 1985

[8] Little, J. D. C, et al., "An *Algorithm for the Traveling Salesman Problem",* O.R.S.A., 11, 972-989, 1963.

[9] Minton, S. et al, *"Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method",* Proc. 8th. Nat. Conf. on A.I. (AAAI-90), 17-24, 1990.

[10] Turner, J. S., *"Almost All k-Colorable Graphs are Easy to Color",* Journal of Algorithms, 9, 63-82, 1988.