Local Search and the Number of Solutions

David A. Clark[*], Jeremy Frank[†], Ian P. Gent[*], Ewan MacIntyre[*], Neven Tomov[‡], Toby Walsh[§]

Abstract

There has been considerable research interest into the solubility phase transition, and its effect on search cost for backtracking algorithms. In this paper we show that a similar easy-hard-easy pattern occurs for local search, with search cost peaking at the phase transition. This is despite problems beyond the phase transition having fewer solutions, which intuitively should make the problems harder to solve. We examine the relationship between search cost and number of solutions at different points across the phase transition, for three different local search procedures, across two problem classes (CSP and SAT). Our findings show that there is a significant correlation, which changes as we move through the phase transition.

Keywords: **computational complexity, constraint satisfaction, propositional satisfiability, search**

1 Introduction

Local search has been proposed as a good candidate for solving the "hard" but soluble problems that turn up at the phase transition in solubility for satisfiability and constraint satisfaction problems. The position of such a phase transition appears to be strongly determined by the expected number of solutions [20, 18, 8, 9]. Recent theoretical analysis has shown that large variances in the number of solutions can occur at the phase transition [11]. In addition, empirical analysis has shown phase transitions can occur when the expected number of solutions is significantly larger than 1 [18]. These results may be important for understanding performance of local search procedures as they might be expected to be strongly influenced by the number of solutions. If there are large number of solutions, local search may stumble on one

[*]Department of Computer Science, University of Strathclyde, Glasgow G1 1XH, Scotland. Tel +44 (141) 552-4400 Fax +44 141 553 4101 E-mail: {dac,ipg,em}@cs.strath.ac.uk

[†]Department of Computer Science, University of California at Davis, Davis, CA. 95616, USA. Tel +1 (916) 758-5925 E-mail frank@cs.ucdavis.edu

[‡]Department of Computing & Electrical Engineering, Heriot-Watt University, Edinburgh EH14 4AS Scotland. Tel +44 131 449 5111 Fax +44 131 451 3431 E-mail neven@cee.hw.ac.uk

[§]IRST, I38100 Trento & DIST, I16145 Genova, Italy. Tel: +39 461 314438. Fax: +39 461 810851. E-mail: toby@itc.it

easily. On the other hand, local search may also be led in conflicting directions by different solutions.

In this paper we show that, across a range of problem classes and local search procedures, the hardest problems occur (as for complete systematic algorithms) at the phase transition in solubility. This is despite the fact that problems beyond the phase transition can have fewer solutions. Problem difficulty across the phase transition is, however, affected by the number of solutions. We identify a correlation between number of solutions and problem hardness for local search. We show this correlation is robust across problem class and types of local search procedure, and across the phase transition. The number of solutions is not the only factor since we identify significant variation in problem hardness when this is held constant. These results are likely to be of considerable importance for understanding phase transition behaviour in local search procedures and for benchmarking such procedures.

2 Background

2.1 SAT

Propositional satisfiability (or SAT) is the problem of deciding if there is an assignment of truth values for the variables in a propositional formula that makes the formula true using the standard interpretation for logical connectives. We will consider SAT problems in conjunctive normal form (CNF); a formula, Σ in CNF is a conjunction of clauses, where a clause is a disjunction of literals, and a literal is a negated or un-negated variable. In k-SAT problems, all clauses contain exactly k literals. Both SAT and k-SAT (for $k \geq 3$) are NP-complete [6]. As is usual [14], we will generate random k-SAT problems with n variables and l clauses, by picking k variables out of the n possible for each clause, and then negating each variable with probability $\frac{1}{2}$.

2.2 CSP

A constraint satisfaction problem (CSP) consists of a set of n variables and a set of constraints. Each each variable v has a domain, M_v of size m_v. Each k-ary constraint restricts a k-tuple of variables, $(v_1, ... v_k)$ and specifies a subset of $M_1 \times ... \times M_k$, each element of which are values that these variables cannot simultaneously take. We consider here binary CSPs in which constraints are only between pairs of variables. As in previous studies [18, 8], we will generate binary CSPs with n variables each with domain m, constraint density p_1, and constraint tightness p_2, by picking exactly $p_1 n(n-1)/2$ out of the $n(n-1)/2$ possible binary constraints between variables. For each selected constraint, we disallow exactly $p_2 m^2$ of the m^2 possible pairs of values of the two variables.

2.3 Phase transitions

Many NP-complete problems like satisfiability and constraint satisfaction, display a rapid transition in solubility as we increase the constrainedness of random problem instances. This phase transition is associated with problems which are typically hard to solve for backtracking procedures [2]. Problems that are under-constrained tend to have many solutions. It is usually therefore very easy to guess one of the solutions. Problems that are over-constrained tend not to have any solutions. As there are many constraints, any possible solution is usually quickly ruled out. At an intermediate point, problems are critically constrained: out of a random sample some will be soluble and some not, and it is usually hard to either find a solution or to prove that none exists. Many investigations have studied phase transition behaviour in backtracking algorithms, in problems such as SAT (e.g. [14, 3]), CSP [8, 16, 18], Hamiltonian circuits [2, 5], and the travelling salesman problem (e.g. [9]).

A uniform treatment of phase transitions in combinatorial problems has recently been presented [9], formalising the notion of 'constrainedness'. Given an ensemble of problems, the constrainedness is defined by,

$$\kappa \quad =_{\text{def}} \quad 1 - \frac{\log_2(< Sol >)}{\mathcal{N}} \tag{1}$$

where $< Sol >$ is the expected number of solutions for a problem in the ensemble, and $\mathcal{N}$ is the number of bits needed to write down a solution, i.e. the base 2 logarithm of the size of the state space. κ lies in the range $[0, \infty)$. If $\kappa \ll 1$ problems are under-constrained and likely to soluble, if $\kappa \gg 1$ problems are over-constrained and likely to be insoluble, and if $\kappa \approx 1$ problems are critically constrained and may be soluble or insoluble.

As in [9] we plot most of our results against the constrainedness, κ of problem instances. This allows phase transitions in different classes such as SAT and CSP to be directly compared. Furthermore, such comparisons are directly related to the number of solutions since,

$$\log_2(< Sol >) \quad = \quad \mathcal{N}(1 - \kappa) \tag{2}$$

For random k-SAT problems, $\mathcal{N} = n$ and $\kappa = -\log_2(1 - 2^{-k})l/n$ [9]. This is a constant multiplied by the familiar parameter l/n [14]. For random CSPs, $\mathcal{N} = n \log_2(m)$ and $\kappa = \frac{n-1}{2} p_1 \log_m(\frac{1}{1-p_2})$ [8].

Despite the extensive literature of phase transitions in backtracking search, there has been little analysis of phase transitions in local search. This is perhaps because phase transition behaviour is usually associated with the transition from soluble to insoluble problems, and local search procedures can only solve soluble problems. It might therefore appear that phase transitions will not be observed with local search procedure. We can, however, conduct experiments on the *soluble phase* of the ensemble. By 'soluble phase', we mean those problems in an ensemble which are soluble, no matter what the generation parameters are. To study the soluble phase, we generated problems at random as described above, and then used a complete backtracking algorithm to eliminate insoluble problems from the ensemble.

2.4 Local search procedures

Local search procedures start with an initial assignment of values to the variables. They then explore their "local neighbourhood" for "better" assignments. The local neighbourhood usually consists of those assignments where the value of one variable is changed. A "score" function is applied to determine which neighbour to move towards. In SAT, we use the number of satisfied clauses. In CSP, we use the number of satisfied constraints. Hill-climbing is used in the GSAT and min-conflicts procedures to maximize the score. Other procedures use more complex procedures for selecting neighbours. For example, the MC-LOG procedure chooses a neighbour probabilistically according to the relative ranking of the scores. Local procedures can, of course, be trapped in local maxima. Various techniques have been developed to overcome this. For example, GSAT simply restarts from a new point in the state space. By comparison, procedures like MC-LOG and simulated annealing allow score-decreasing moves with a certain probability. The experiments in this paper use three local search procedures: GSAT, a CSP analogue of GSAT called GCSP, and a min-conflicts algorithm for CSPs called MC-LOG.

GSAT [17] is a local search procedure for SAT which begins with a random generated initial truth assignment, then hill-climbs by reversing or "flipping" the assignment of the variable which increases the number of satisfied clauses the most. After a fixed number, MaxFlips, of moves, search is restarted from a new random truth assignment. Search continues until we find a model or we have performed a fixed number, MaxTries, of restarts.

GCSP [19] is an analogous procedure to GSAT for CSPs. It begins with a random generated assignment of values to variables, then hill-climbs by finding a new variable-value assignment which increases the number of satisfied constraints the most. After a fixed number, MaxChanges, of moves (exactly analogous to MaxFlips in GSAT), search is restarted from a new random assignment. Search continues until we find a solution or we have performed a fixed number, MaxTries, of restarts.

MC-LOG is based on min-conflicts hill-climbing [13] but with an ability to escape local minima. Unlike GCSP, MC-LOG does not consider all variables, but instead selects randomly a variable in conflict with some constraint. The local neighbourhood for this variable consists of alternative values for it. Unlike min-conflicts hill-climbing, MC-LOG does not select the neighbour which minimizes the number of conflicts, but ranks all neighbours according to their min-conflicts 'score', and selects one probabilistically. Changing the value of this variable is called a 'repair'. The selection function is logarithmic, so the 'best' value is chosen most often, but not exclusively as with min-conflicts*. The number of conflicts can therefore occasionally increase, enabling the procedure to escape from local maxima.

*More precisely, we pick the ith ranked value where $i = \text{int}(\log_2(1/r)/w)$ and r is a random number in $[0, 1]$ and w is some fixed weighting.

3 Phase transitions and local search

We first investigate the performance of local search as we vary the number of solutions for a fixed size of problem. Naively, one might think that problems will get monotonically harder as we decrease the number of solutions since we must search for an ever smaller number of needles in a haystack. However, even though all the problems tested are soluble, behaviour is affected by the solubility phase transition. Indeed, the hardest problems for local search seem to occur at the same point as the hardest problems for complete search, namely at the phase transition in solubility. In this paper, we take this to be the point in the phase space where 50% of problems are soluble and 50% insoluble. This point is often associated with the hardest mean search cost for backtracking algorithms [3].

3.1 MC-LOG

In Fig 1 (left), we present results for MC-LOG on 1000 soluble CSPs with $n = 20$, $m = 10$, and $p_1 = 0.5$, and p_2 varying from 0.32 to 0.42, corresponding to a range of κ from 0.80 to 1.12. We plot search cost (the number of repairs) against the constrainedness, κ. The phase transition in solubility starts at $\kappa = 0.89$ where 99.1% of problems generated are soluble[†], the nearest point to 50% solubility is at $\kappa = 0.95$ where 57.0% are soluble, and our graphs extend to regions where very few problems are soluble. At $\kappa = 1.12$, only 1.2% of problems had solutions. The peak in median search cost is at $\kappa = 0.99$ while the peak in mean search cost is slightly earlier at $\kappa = 0.95$. Surprisingly, at larger values of κ, the search cost decreases even though the average number of solution is declining. As with complete procedures, the peak average search cost is associated with the solubility phase transition.

Similar results are obtained if we study CSPs with different constraint densities. In Fig 1 (right), we vary p_2 and plot median search cost for $p_1 = 0.30$ to $p_1 = 1.00$, i.e. complete constraint graphs. As p_1 increases the phase transition occurs at smaller values of p_2. In all cases the peak median search cost is within 0.01 of the value of p_2 where 50% of problems are soluble. For comparison of different problem classes, we also plot our data against κ instead of p_2, as in Fig 1. As the constraint density, p_1 increases the peak search cost (and solubility transition) occurs nearer to the expected value of $\kappa = 1$. As p_1 increases, the mean search cost at the phase transition increases, and by (2), the expected number of solutions decreases. This suggests a correlation between the average number of solutions and search cost at the phase transition. However the picture is not clear, since at a fixed κ there is a fixed expected number of solutions, yet the search cost varies by a large factor for different p_1.

We draw two conclusions from this data. First, we observe an 'easy-hard-easy' pattern of problem difficulty. The hard region is associated with the solubility phase transition, despite the fact that as we make problems more constrained, they have fewer solutions. This is consistent with the results of [14, 18] for backtracking algorithms applied to soluble problems. Second, there is some correlation between peak search cost and expected number of solutions at the phase transition.

[†] Recall, that we simply discard insoluble problems.

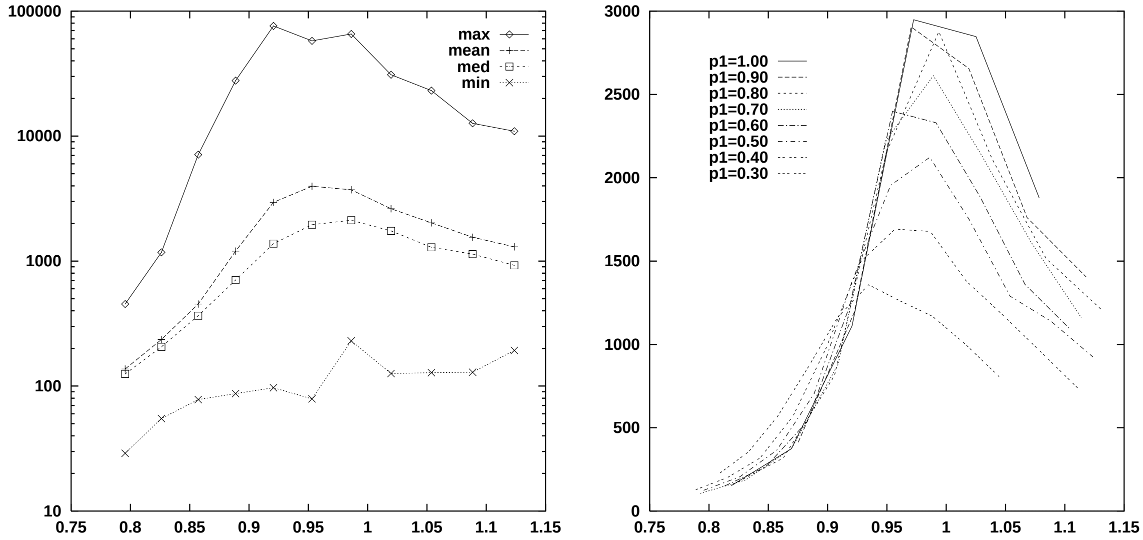

Figure 1: (left) Search cost for MC-LOG on (20,10,0.5) problems as p_2 varies, plotted against kappa. (right) Median search cost for MC-LOG on (20,10,p_1) problems.

3.2 GCSP

To determine if the results of Section 3.1 are specific to the MC-LOG procedure, we re-ran the experiment reported in Fig 1 (left) using the local search procedure GCSP. To minimise variation, we tested GCSP with the identical problems used with MC-LOG. We set MaxChanges to 500 and ran until problems where solved, i.e. we effectively set Max-Tries to infinity. We plot total changes used by GCSP against κ in Fig 2 (left). Total changes is calculated as MaxChanges times the number of failed tries, plus the number of Changes on the final try, and gives a measure of search cost for GCSP.

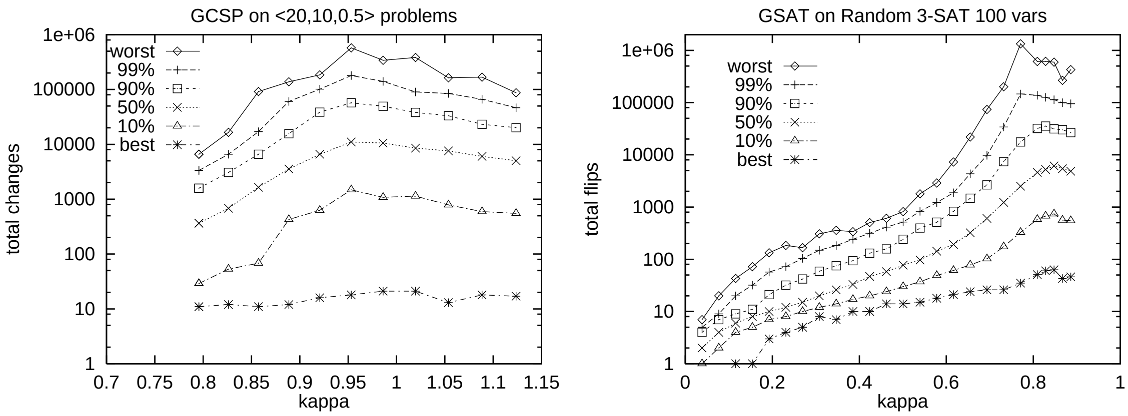

Figure 2: (left) GCSP phase transition behaviour (right) GSAT phase transition behaviour

Behaviour is broadly similar to that seen with MC-LOG even though these procedures have significant differences. GCSP uses restarts instead of probabilistic acceptance to avoid local maxima. In addition, GCSP makes a global choice of the best variable-value assignment rather than a local choice of value for a selected variable. This suggests that other local search procedures for binary CSPs will display an easy-hard-easy pattern with the peak in search cost occuring at the solubility phase transition.

3.3 GSAT

It is likely that these results for CSPs will apply to other NP-complete problem classes like SAT. To test this hypothesis, we performed a similar experiment with 100 variable random 3-SAT problems. We varied the number of clauses from 20 to 420 in steps of 20, and from 420 to 460 in steps of 10. The end point corresponds to a value of $\kappa \approx 0.89$. At each point we tested GSAT on 1000 soluble problems. We set MaxFlips to 342, the optimal value for the middle of the phase transition reported by [10]. As a measure of search cost we use the total flips used, calculated analogously to total changes in GCSP. The phase transition in solubility starts at 380 clauses, $\kappa \approx 0.73$ where 99.3% of problems generated are soluble. The point nearest 50% solubility is 430 clauses, $\kappa \approx 0.83$, where 48.2% of problems were soluble. At the end point of our experiment, 9.9% of problems were soluble. The fact that κ is smaller at the phase transition than in previous graphs indicates that there are more solutions at this point than for comparably sized CSPs.

Fig 2 (right) shows different percentiles of search cost plotted against κ. The peak in median is at the next point beyond the solubility transition, and indeed the peak in the 90th percentile of search cost is at 430 clauses, the 50% solubility point. Even best case behaviour seems to get easier as we increase κ.

To conclude, for both random CSPs and SAT problems, we see an easy-hard-easy pattern in the cost of local search procedures with the peak in search cost occuring at the phase transition in solubility.

4 Search cost and number of solutions

In the last section, we suggested that naively one expects problems to get monotonically harder for local search as the number of solutions decreases. To determine how true this is, in Fig 3 we give scatter plots of search effort for MC-LOG ($\log_{10}$(repairs) to find a solution) against $\log_{10}$(number of solutions). Each data point reported is the mean of 10 runs of MC-LOG to solve an individual problem.

In Fig 3 (left), we give a scatter plot for all problems in the soluble phase. At large numbers of solutions, this is a close correlation between the number of solutions and the search cost, and the spread in search costs is relatively small. The overall shape of Fig 3 (left) suggests a linear correlation between the log number of solutions and log search cost. We performed linear regression on this data, finding a best fit gradient of -0.31 with a correlation coefficient r of -0.79. At small number of solutions, however, the spread is huge, up to nearly 3 orders of magnitude. This suggests that search cost is not simply a function of the number of solutions.

We obtain a better picture of behaviour if we look at fixed points in the phase space. In Fig 3 (right), we give a scatter plot for problems with a low constraint tightness ($p_2 = 0.32$), in Fig 4 (left), problems from the middle of the phase transition ($p_2 = 0.37$), and in Fig 4 (right), problems with a large constraint tightness ($p_2 = 0.42$). At each point, there is less overall spread in the search cost for a given number of solutions than seen in Fig 3 (left). We performed regression analysis at each point separately, and the resulting lines are shown. At $p_2 = 0.32$ we estimated a gradient of -0.36 with $r = -0.56$, at $p_2 = 0.37$, we estimated a gradient of -0.61 with $r = -0.70$, and at $p_2 = 0.42$, we estimated a gradient of -0.29 with $r = -0.28$. Notice that the gradient is steepest at the solubility phase transition. Problems with a low constraint tightness have a large number of solutions and search cost is relatively uniform and small. At the phase transition, there is a large variation in the number of solutions. For problems with few solutions at the phase transition, search cost tends to be large. Although problems with a larger constraint tightness have a smaller number of solutions, search cost for problems with the same number of solutions tends to be less than at the phase transition. As seen earlier, the overall cost is greatest at the phase transition. Although there is still considerable variability, the hardest problems are those from the phase transition with a few solutions.

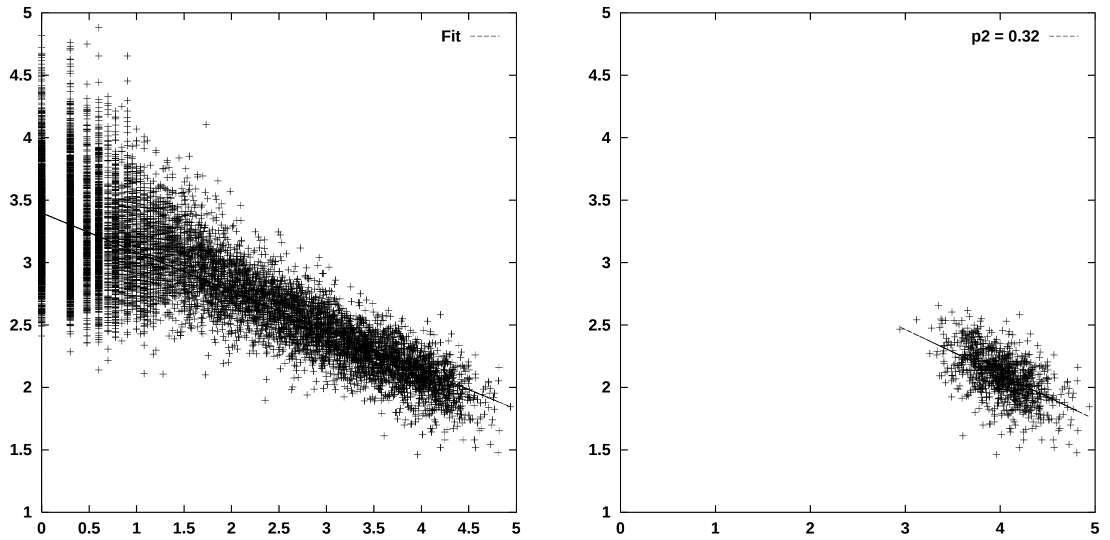

Figure 3: $\log_{10}(repairs)$ for MC-LOG (y-axis) plotted against $\log_{10}(solutions)$ (x-axis). (left) All p_2 , (right) $p_2 = 0.32$

4.1 GCSP

To determine if these results are specific to the MC-LOG procedure, we also made scatter plots for the logarithm of search cost of GCSP against the logarithm of the number of solutions. Each data point is the cost of solving an individual problem once. Fig 5 (left) gives the plot for the middle of the phase transition. The regression line has gradient -0.56 and $r = -0.40$. Results are very similar to MC-LOG. Again there is considerable variability, problems with more solutions are generally much easier than problems with very few solutions. The hardest problems are again those from the phase transition with a single or very few solutions. The greater noise in

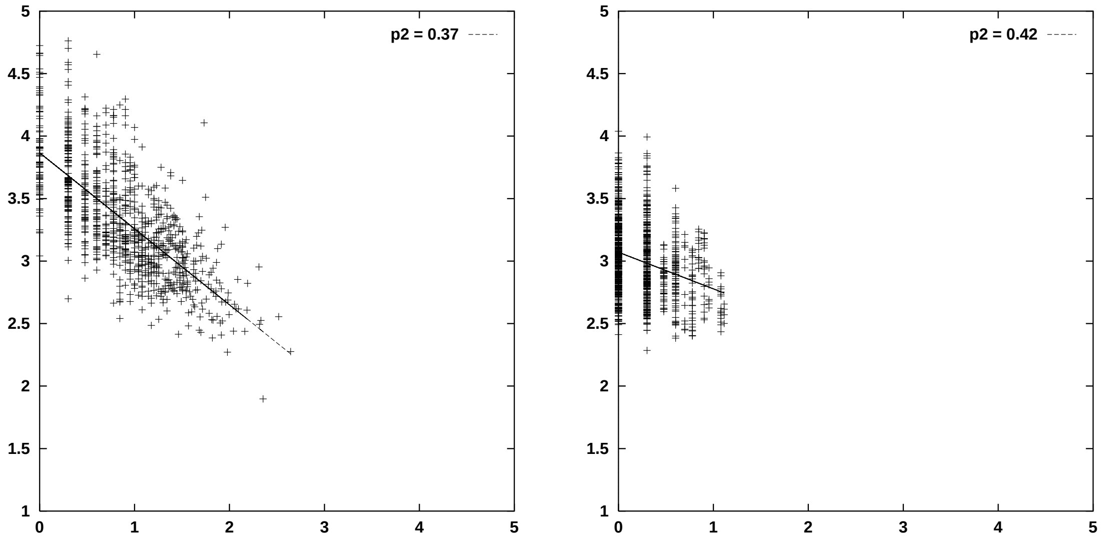

Figure 4: $\log_{10}(repairs)$ for MC-LOG (y-axis) plotted against $\log_{10}(solutions)$ (x-axis). (left) $p_2 = 0.37$, (right) $p_2 = -0.42$

this figure than seen with MC-LOG is probably due to only reporting a single rather than an average of 10 runs.

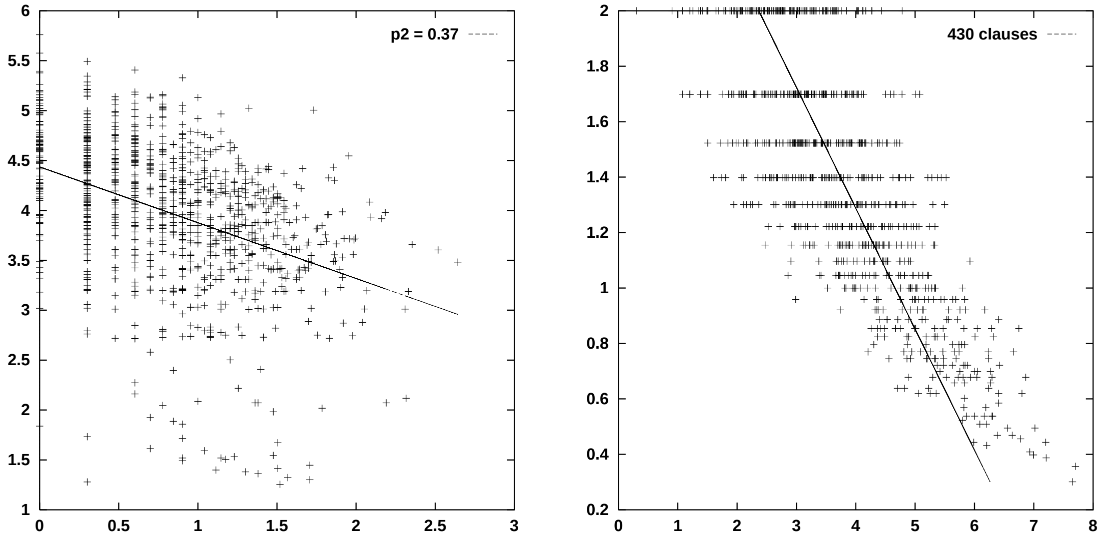

Figure 5: $\log_{10}(cost)$ (y-axis) plotted against $\log_{10}(solutions)$ (x-axis) (left) GCSP at $p_2 = 0.37$ (right) GSAT at $n = 100$, $l = 430$

4.2 GSAT

Once again, we wished to see if our results extend beyond CSPs to SAT. To test this, we randomly generated 1000 soluble 3-SAT problems with 100 variables and 430 clauses, the point nearest the solubility phase transition as reported above. We ran GSAT on 100 tries and used this to give an estimate of search cost as simply 100 divided by the number of successes for each problem.[‡] We also found the exact number of solutions that each problem has. The relationship between these two measures is shown in Fig 5 (right). Again, there is a strong tendency for problems with more solutions to be easier, although again there is a lot of noise. We modelled this by a regression line with gradient -0.44 and $r = -0.77$. The granularity in the search cost in the figure is due to the measure we used, and the use of this measure may make the regression less accurate. We intend further experiments to investigate this.

To summarise, we have shown a strong if noisy correlation between search cost and number of solutions. The correlation seems to be linear on a log-log scale. The dependency between search cost and number of solutions changes as we change the constrainedness of problems, with the hardest problems being those at the solubility phase transition with very few solutions.

5 Regression analysis through phase transitions

We have shown that the correlation between search cost and the number of solutions varies at different points in the solubility phase transition. We now look more closely at this variation.

Fig 6 shows the regression lines for MC-LOG for all values of constraint tightness p_2 across the phase transition. There is a distinct pattern in these regression lines; at low constraint tightness, the magnitude of the gradient is low, but increases to reach a peak when $p_2 = 0.36$ ($\kappa = 0.92$), which is at the phase transition. The gradient then decreases as constraint tightness increases. This is shown more clearly in Fig 7 (left), which shows the values of the gradient against κ. This makes clear that the steepest gradient is at the solubility phase transition. We interpret this as suggesting that the number of solutions has most influence on search cost at the solubility phase transition.

Fig 7 (right) shows how the regression fit gradient changes against κ, for $p_1 = 0.30$ to $p_1 = 1.00$. In each case, the minimum gradient (maximum absolute gradient) occurs at or very close to the 50% solubility point on the phase transition. Once again, this suggests that the number of solutions has most influence on search cost at the phase transition.

5.1 GCSP

We again continued our investigations by seeing if our results would also be seen in GCSP. Accordingly, we plotted the regression gradients obtained from analysis of experiments on problems from the classes (20,10,0.5) with varying p_2. The results are seen in Fig 8 (left). The same pattern emerges as in Fig 7 (left). The steepest gradient is at exactly the same point. Some noise can be seen in this graph, perhaps due as we noted earlier to not averaging GCSP results over many runs. Nevertheless, it seems that for a range of CSP local search methods, the number of solutions matters most at the phase transition.

[‡]Thus we are using a different cost measure to that used above for GSAT in Fig 2.

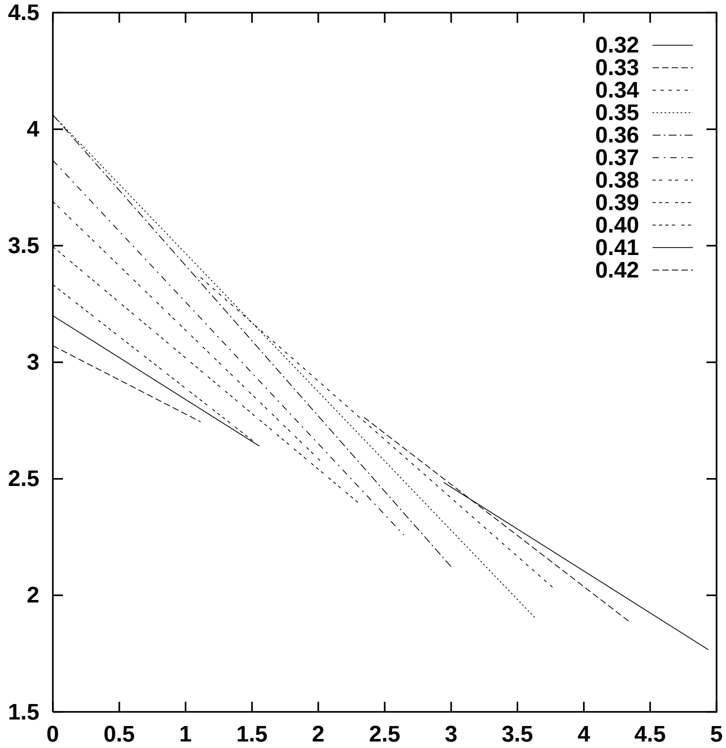

Figure 6: Regression fits for log search effort (x-axis) vs log solutions (y-axis), as p_2 changes (20, 10, 0.50, p_2)

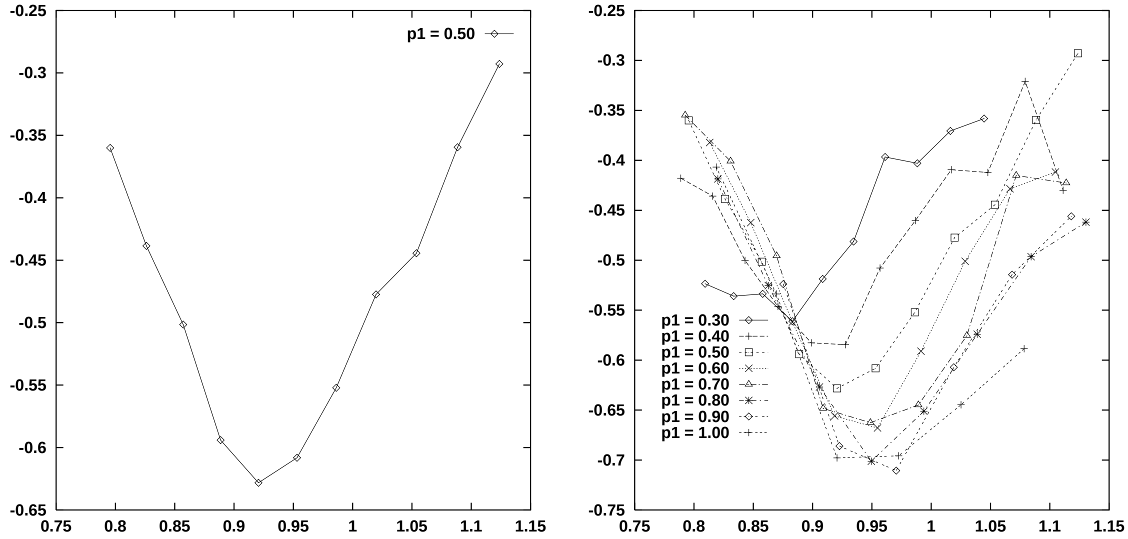

Figure 7: Regression fit gradients (y-axis) plotted against κ (x-axis) for (left) $p_1 = 0.50$, and (right) for p_1 from 0.30 to 1.00.

5.2 GSAT

To conclude our investigations of how regression gradients changed through the phase transitions, we analysed our data for GSAT with 100 variables from 410 to 480 clauses. Again the gradient of the regression between number of solutions and search cost changes as we vary the constrainedness of problems. Recalling that the solubility phase transition is at 430 clauses, in this case we do not find the steepest gradient at this point. As yet we are unclear as to the reasons for this change from our results on CSPs. In particular, it must for the moment remain open whether this is due to an inherent difference between 20 variable CSP instances and 100 variable 3-SAT instances, or to some more mundane feature such as the differing measures of search costs used. One significant difference between the CSP and 3-SAT classes we have studied here may be that the solubility transition in 3-SAT occurs at lower values of κ, i.e. where more solutions occur in relation to problem size.

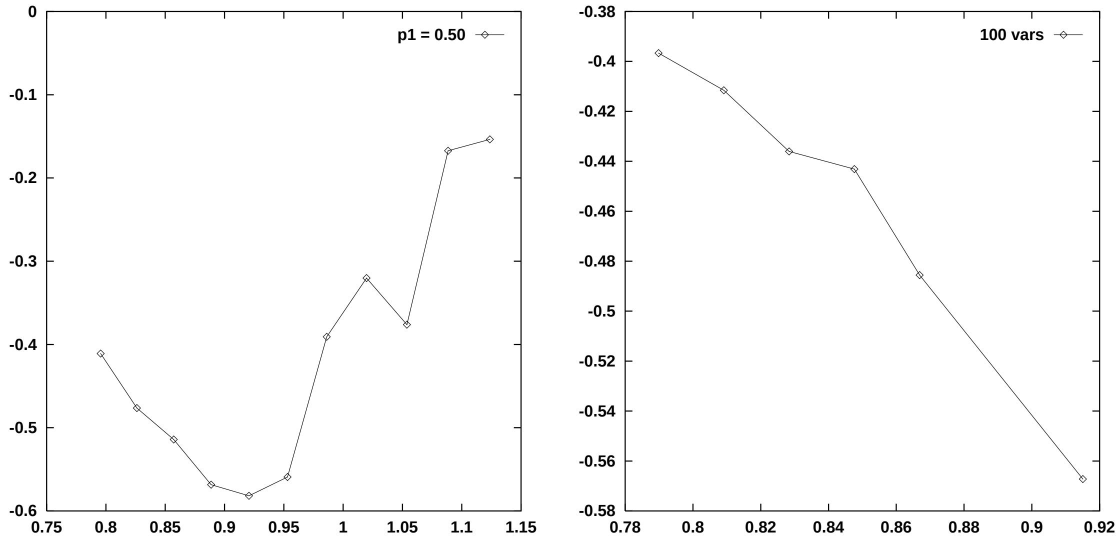

Figure 8: Regression fit gradients (y-axis) plotted against κ (x-axis). (left) GCSP on (20,10,0.5) problems. (right) GSAT on $n = 100$ problems

6 Comparison with complete algorithms

Systematic backtracking procedures will also be affected by the number of solutions. If there are many solutions, then it may not to be hard to find a path that leads to one. If there are few, it may be hard to find a path that ends in a solution. In Fig 9, we give scatter plots for the search effort for the FC-CBJ-BZ algorithm (measured in consistency checks) against the number of solutions. We give plots for both the search effort to find the first solution and to find all solutions. FC-CBJ-BZ is a forward checking algorithm with conflict-directed backjumping [15] using the Brelaz heuristic for dynamic variable ordering [1]. This is currently one of the best algorithms for CSPs.

There is a tendency for search cost to find the first solution to decrease with the number of solutions, and this can be regressed by a line with gradient -0.18 and $r = -0.21$. However the tendency is very weak and even noisier than the plots seen earlier for local search. For finding all solutions, there seems to be no tendency for increasing numbers of solutions to reduce cost. This is not surprising since to find all solutions, all parts of the search space must be explored in full. It seems that new features of the solution space will have to be investigated to predict the search cost for complete algorithms.

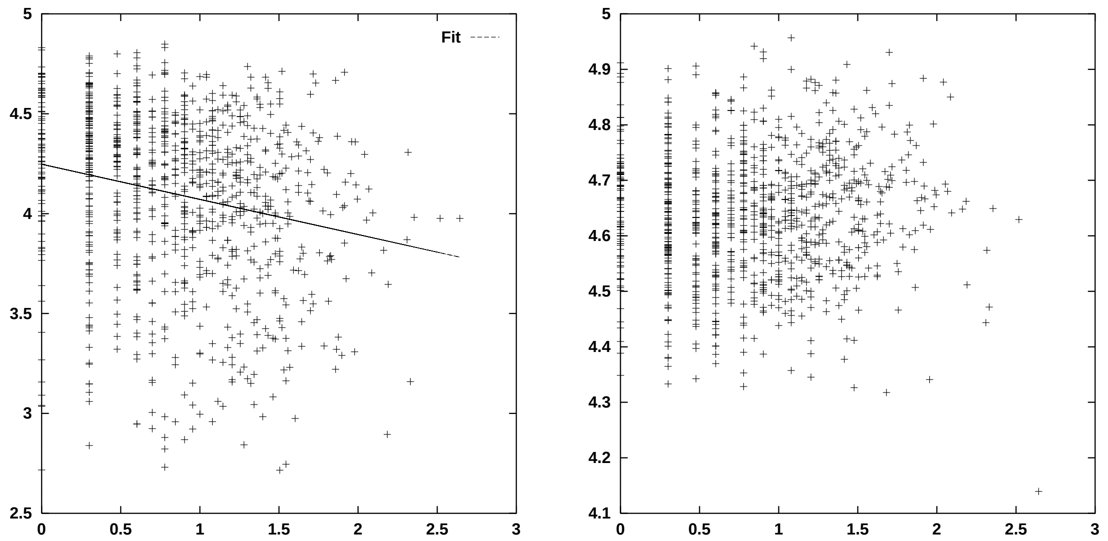

Figure 9: $\log_1 0(checks)$ (y-axis) for FC-CBJ-BZ against $\log_1 0(solutions)$ (x-axis). (left) checks until first solution found (right) is checks to find all solutions

7 Related Work

Phase transitions in backtracking algorithms have been the subject of enormous study in recent years, but comparatively little attention has been paid to the behaviour of local search algorithms as the position in the phase space changes. Experimental results for GENET and GSAT applied to random 3-colouring instances are reported by [4]. These results are consistent with those reported in Section 3 above, in that search cost reduces beyond the phase transition, and further suggest that the results reported here are unlikely to be dependent on the particular algorithms or classes studied.

Gent and Walsh showed that a range of variants of GSAT applied to non-random problems gave good performance on problems with many solutions such as the n-queens problem, and poor performance on problems with few solutions, for example quasigroup construction problems [10]. They speculated that the number of solutions in relation to problem size would be critical in understanding local search cost. In this paper we have shown that this speculation is confirmed when random problems are studied, although there are other features which must be taken into account such as position in the phase space. This still does not yield a full explanation of behaviour, so we wish to research the topology of local search in more depth, following studies such as [7, 12].

8 Conclusions

We have investigated in depth the relationship between the number of solutions and search cost for local search procedure. Although there is no single simple story (for example, search cost is inversely proportional to the solution density), we have identified some important connections. In particular, the hardest problems tend to have few solutions and usually occur (as with complete, systematic algorithms) at the solubility phase transition. We have shown that there is a significant correlation between the number of solutions and problem hardness for local search. This correlation is robust across problem class and types of local search procedure, and across the phase transition. The number of solutions is, however, not the only factor determining problem hardness since there is significant variation in problem hardness when the solution density is held constant. These results improve our understanding of phase transition behaviour and of the factors affecting the performance of local search methods.

References

[1] D. Brelaz. New methods to color the vertices of a graph. *Comms. ACM*, 22:251–256, 1979.

[2] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proceedings of the 12th IJCAI*, pages 331–337. International Joint Conference on Artificial Intelligence, 1991.

[3] J.M. Crawford and L.D. Auton. Experimental results on the crossover point in satisfiability problems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 21–27. AAAI Press/The MIT Press, 1993.

[4] A. Davenport. A comparison of complete and incomplete algorithms in the easy and hard regions. In *Proceedings, Workshop on Studying and Solving Really Hard Problems, CP-95*, pages 43–51, 1995.

[5] J. Frank and C. Martel. Phase transitions in random graphs. In *Proceedings, Workshop on Studying and Solving Really Hard Problems, CP-95*, 1995.

[6] M. R. Garey and D. S. Johnson. *Computers and intractability : a guide to the theory of NP-completeness.* W H Freeman, 1979.

[7] I. P. Gent and T. Walsh. An empirical analysis of search in GSAT. *Journal of Artificial Intelligence Research*, 1:47–59, September 1993.

[8] I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. Scaling effects in the CSP phase transition. In *Principles and Practice of Constraint Programming*, pages 70–87. Springer, 1995.

[9] I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The constrainedness of search. Tech. rep., IRST, Trento, 1996. Under review for AAAI-96.

[10] I.P. Gent and T. Walsh. Unsatisfied variables in local search. In *Hybrid Problems, Hybrid Solutions*, pages 73–85. IOS Press, 1995. Proceedings of AISB-95.

[11] A. Kamath, R. Motwani, K. Palem, and P. Spirakis. Tail bounds for occupancy and the satisfiability threshold conjecture. *Randomized Structure and Algorithms*, 7:59–80, 1995.

[12] S.A. Kauffman. *The Origins of Order*. Oxford University Press, 1993.

[13] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *AAAI-90, Proceedings Eighth National Conference on Artificial Intelligence*, pages 17–24. AAAI Press/MIT Press, 1990.

[14] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proceedings, 10th National Conference on Artificial Intelligence*, pages 459–465. AAAI Press/The MIT Press, 1992.

[15] P. Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9:268–299, 1993.

[16] P. Prosser. Binary constraint satisfaction problems: Some are harder than others. In *Proceedings of ECAI-94*, pages 95–99, 1994.

[17] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings, 10th National Conference on Artificial Intelligence*. AAAI Press/The MIT Press, 1992.

[18] B.M Smith. Phase transition and the mushy region in constraint satisfaction problems. In *Proceedings of ECAI-94*, pages 100–104, 1994.

[19] N. Tomov. Hill-climbing heuristics for solving constraint satisfaction problems. 4th year project report, Department of Artificial Intelligence, University of Edinburgh, 1994.

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