An empirical investigation into the exceptionally hard problems

Andrew Davenport and **Edward Tsang**
Department of Computer Science, University of Essex,
Colchester, Essex CO4 3SQ, United Kingdom.
{daveat,edward}essex.ac.uk

Abstract

Recently "exceptionally hard" problems have been found in the easy region of problem spaces which can be orders of magnitude harder to solve than even the hardest problems in the phase transistion. However there is some uncertainty over whether this phenomena of exceptionally hard problems occurs for all algorithms, occurs only for complete algorithms or is purely algorithm dependent. The purpose of this paper is to assess the performance of a range of algorithms on problems in the easy region in order to address this issue. We present results of an empirical investigation which show that both soluble and insoluble exceptionally hard problems can occur for even very sophisticated complete search algorithms, although the likelihood of such problems occuring varies significantly depending on the algorithm being used to solve them. GENET, an incomplete, iterative repair-based search did not encounter any soluble, exceptionally hard problems in the easy region.

Introduction

Cheeseman et al [2] have shown that NP-complete problems display a phase transistion between regions of under-constrained, mostly soluble problems and regions of over-constrained, mostly insoluble problems as some parameter is varied. The under-constrained problems tend to be relatively easy to solve and thus form an "easy" region of the problem space. Similiarly the over-constrained problems tend to be relatively easy to prove insoluble. However the peak in median search cost is found in the phase transistion, where we find both soluble and insoluble problems. These problems have been typically found to be hard to solve by all algorithms and thus can be characterized as being inherently hard.

Recently problems have been found in the easy region of problem spaces which can be orders of magnitude harder to solve than even the hardest problems in the phase transistion [4, 7, 11]. Williams and Hogg [7] have found that the hardest problems are concentrated below the phase transistion peak in median search cost and identify these problems with a second phase transistion, corresponding to the transistion between polynomial and exponential scaling of the average search cost. Gent and Walsh [4] observe that the greatest variability in problem solving cost occurs in this region and it occurs for both soluble and insoluble problems.

There is some uncertainty over whether the phenomena of exceptionally hard problems occurs for all algorithms, only occurs for all complete algorithms or is purely algorithm dependent. Smith [11] concludes that some exceptionally hard problems are "inherently

more difficult than other similiar problems, independently of the number of solutions they have". Gent and Walsh [4] make the weaker conjecture that the phenomena of exceptionally hard problems may occur for all complete algorithms. It is interesting to note that nobody has yet reported any exceptionally hard, soluble problems for incomplete, local search. Williams and Hogg [7] have looked at the performance of heuristic repair on graph colouring problems while Gent and Walsh [5] have examined the performance of GSAT on easy 3-SAT problems, but neither found any exceptionally hard problems for these algorithms. Does this mean that there are no exceptionally hard problems for local search ? If this is the case should we always prefer local search to complete search when tackling problems in the easy region, or can we avoid exceptionally hard problems for complete search by using more sophisticated algorithms ? As Prosser [10] puts it, are exceptionally hard problems merely existence proofs for exceptionally bad algorithms ?

In this paper we try to go some way towards tackling these issues. In the sections which follow we present results of an empirical study comparing a number of well-known constraint satisfaction algorithms on problems in the easy region.

Problem generation

We ran some preliminary experiments in order to determine for which types of problems exceptionally hard problems are most likely to occur. Experiments on randomly generated binary constraint satisfaction problems showed that exceptionally hard problems are more likely to occur when there are many variables with small domains, and the constraint graph of the problem is sparse. Hence we decided to use graph 3-colourability problems.

A graph colouring problem consists of a graph and a maximum number of colours, where the goal is to assign one colour to each node such that no two adjacent nodes in the graph are assigned the same colour. We use the connectivity of the graph as a parameter to distinguish between easy and hard regions of the space of graph colouring problems [2].

Empirical evaluation

For all experiments we generated graphs varying in connectivity in steps of 0.1. For the smaller sized graphs (50 and 100 nodes) we varied graph connectivity from 2.0 to 5.0. The critical point of the phase transistion is expected to occur around a connectivity of 4.6 for graph 3-colourability. For larger graphs (200 and 400 nodes) we were only able to obtain results from the easy region, between connectivities of 2.0 and 4.3, within the time available to us.

At each connectivity we generated 1,000 problems. For complete algorithms we require the algorithm to either find the first 3-colouring of the graph or show that there is no 3-colouring. We only ran incomplete algorithms on soluble problems, and required them to find a single 3-colouring. We used forward checking as our base complete search algorithm, and experimented with variable ordering and backjumping strategies[1]. We also experimented with GENET [3], an incomplete, local search algorithm based upon the min-conflicts heuristic [8] but with the ability to escape local minima.

First we looked at graphs consisting of 50 nodes. Figures 1 and 2 presents the results for forward checking with fail first variable ordering (FC-FF), showing the 100%, 99%, 90%, 75% and 50% percentiles for the number of consistency checks required to solve all problems and to solve only the soluble problems. Here the hardest problems for FC-FF, soluble and insoluble, do not occur in the phase

[1] Forward-checking and other algorithms used in this paper are described in [12]. See also [9]

transistion but in a region of relatively low connectivity. In fact the peak in search cost that we would expect in the phase transistion is barely recognizable in this figure. These results correspond to what has been reported in the literature on exceptionally hard problems [4, 7].

We ran forward-checking with conflict-directed backjumping and fail-first variable ordering (FC-CBJ-FF) [9] on the same problem set, and present results for this algorithm in figures 3 and 4. Here we see a significant improvement in performance. The hardest problems are still in the easy region, however these exceptionally hard problems are mostly insoluble. Figure 4 shows that for soluble problems the exceptionally hard instances occur at higher connectivities, closer to the peak in median search cost, than for FC-FF.

Next we looked at using the Brélaz heuristic [1, 13] to order variables instead of the fail-first principle. The Brélaz variable ordering can be regarded as an extension of fail-first—it first selects the unlabelled variables with the smallest domains. In tie situations it selects the one connected to the largest number of unlabelled variables, breaking further ties randomly.

We ran forward-checking with Brélaz on the 50 node problems, and present the results in figures 5 and 6. Although there are exceptionally hard problems for FC-BZ, the cost of solving these exceptionally hard instances when they do occur is less than for FC-CBJ-FF. This suggests that a good variable ordering appears to be more important than the ability to backjump.

We present results for forward-checking with conflict-directed backjumping and Brélaz (FC-CBJ-BZ) in figures 7 and 8. Here the hardest soluble and insoluble problems are tending to occur at higher connectivities than for FC-BZ. In fact, by adding conflict-directed backjumping we have eliminated most of the soluble problems which were exceptionally hard for FC-BZ at the lower connectivities.

Figure 9 shows results obtained for GENET on the soluble instances of these problems. Here we find no problems which could be called exceptionally hard for GENET in the easy region.

These results for 50 node problems are interesting since they show that the occurence of exceptionally hard problems is, at least to some extent, dependent upon the algorithm being used to solve them. This is in contrast to problems in the phase transistion, which appear to be hard for all algorithms. However, Gent and Walsh [4] have observed that "the use of better heuristics appears to delay, but not postpone indefinitely, the appearance of these difficult (exceptionally hard) problems". To test this hypothesis we decided to run the algorithms on larger graphs. Our next problem set was graphs consisting of 100 nodes.

Figures 10 and 11 show results for FC-CBJ-FF on 100 node graphs. The peak in search cost at connectivity of 2.7 was caused by a single, soluble problem. The number of consistency checks required by FC-CBJ-FF to solve this problem was more than was required to prove insoluble the two problems which caused the peak at connectivity of 2.8. We ran FC-CBJ with a natural variable ordering on this graph and found a solution without backtracking. Since we would expect, in general, FC-CBJ-FF to outperform FC-CBJ, we can deduce that the high cost of solving this problem for FC-CBJ-FF was caused by heuristics, in this case fail-first, occasionally failing spectacularly.

The results for FC-BZ on this problem set are given in figures 12 and 13. The peak at connectivity of 2.5 is caused by a soluble problem taking $31,992,299$ consistency checks to solve. FC-CBJ-FF took 341 consistency checks to solve the same problem. FC-BZ has a smaller peak at connectivity 2.7 than

FC-CBJ-FF for soluble problems, however this peak was not caused by the same problem which caused the peak for FC-CBJ-FF at this connectivity.

Next we looked at FC-CBJ-BZ on this problem set (figures 14 and 15). FC-CBJ-BZ encountered no exceptionally hard, soluble problems, although the hardest problem was an insoluble problem at connectivity of 3.6. This was the problem which also caused the peak at connectivity of 3.6 for FC-BZ. However FC-CBJ-FF found this particular problem easy to prove insoluble, requiring just $2,338$ consistency checks, compared to $409,203$ for FC-CBJ-BZ and $1,172,914$ for FC-BZ.

The results for GENET are given in figure 16 for the soluble instances. Once again we did not find any exceptionally hard, soluble problems for GENET.

Figures 17 and 18 show the performance of FC-CBJ-BZ on a sample of 200 node graphs. We still did not find soluble problems which could be called exceptionally hard for this algorithm, although we did find some insoluble ones. The results for GENET are given in figure 19 for the soluble instances.

Our final problem sample was graphs of size 400 nodes (figures 20 and 21). Here for FC-CBJ-BZ we did find a soluble, exceptionally hard problem at connectivity of 3.8. Figure 22 shows results for GENET on the soluble 400 node problems. GENET managed to find solutions for all these problems, without finding any exceptionally hard soluble problems in the easy region. Furthermore the shape of the 100% percentile for GENET appears to be becoming more stable as we increase the size of the graph.

Discussion

Smith [11] identifies a number of possible reasons for the occurence of exceptionally hard problems:

"The problem has no solutions when most problems in the same region of the problem space have many solutions."

We have found insoluble problems in the easy region which can be exceptionally hard to prove insoluble for certain algorithms, but not for others. However we did find exceptionally hard insoluble problems for all the complete search algorithms we looked at.

"The problem has very few solutions."

In this case we would expect these problems to be hard for all algorithms since they must, on average, explore a larger portion of the search space before finding a solution. We did not find any soluble problems in the easy region which were hard for all algorithms. In fact GENET, the incomplete, stochastic search algorithm, did not find any exceptionally hard soluble problems in the easy region.

"The problem may have many solutions but they are clustered together in a limited region of the search space, and, because of the variable and value ordering, the algorithm will not reach any solutions until it has traversed most of the search space."

Smith [11] cites this as the reason why some problems are inherently more difficult than others, independently of the number of solutions they have. The cause of the difficulty is that the algorithm may select a wrong value for a variable early in the search and will not change this value until it has traversed the entire search tree below this variable. Although we did find soluble problems where FC-FF, FC-CBJ-FF and FC-BZ explored almost the whole search space before finding a solution, we did not find any such problems for FC-CBJ-BZ. However such problems may appear for FC-CBJ-BZ if we run it on larger problem samples. Stochastic search algorithms such as GENET do not have this problem since they do not need to be systematic in this way. Thus we would not expect these problems to be hard for such algorithms.

"The search space induced by the algorithm

is exceptionally large, so that, whenever solutions occur, it will take a long time to reach the first one."

Smith backs up this argument with experimental results showing that the search space size can vary quite significantly depending on the constraint graph. However the algorithm used in these experiments was a chronological backtracking algorithm, FC-FF. When backtracking occurs in FC-FF it may not backtrack immediately to the variable which is the cause of the conflict, thus the phenomena of "thrashing" will occur. This is going to result in a much larger search space, and explains the difference in the number of exceptionally hard problems found for FC-FF and FC-BZ which were not hard for FC-CBJ-FF and FC-CBJ-BZ. We believe that by using backjumping rather than chronological backtracking we can significantly improve the performance of an algorithm on these kinds of exceptionally hard problems.

An interesting point to arise out of this is that *in practice* it often appears better to use simple chronological backtracking algorithms such as FC-FF rather than FC-CBJ-BZ since, for most easy problems, the *median* CPU time to find a solution is lower. However, the presence of rare, exceptionally hard problems in large samples of problems results in the *mean* CPU time for FC-CBJ-BZ being lower than that for FC-FF. Thus investing more time in variable ordering and retaining backjumping information will pay off in the long run.

The fact that GENET managed to find solutions to all the soluble problems very quickly without finding any exceptionally hard problems would indicate that, in practice, local search may be preferable to complete search for solving very large problems in the easy region. It should be possible to find complete search algorithms which are better than FC-CBJ-BZ in solving exceptionally hard problems, for instance dynamic backtracking [6] or algorithms which use arc-consistency lookahead. Similarly it may be possible to improve on the Brélaz variable ordering heuristic, since at some point it still effectively makes a random choice amongst variables. However we found that GENET already outperforms FC-CBJ-BZ in terms of CPU time, and improving on FC-CBJ-BZ would make this comparison worse. In terms of sacrificing completeness, this becomes less of an issue with large problems. We have found that insoluble problems in the easy region become rarer as problems becomes larger, and when these problems do occur they may take too long to prove insoluble in practice.

Conclusions

We have presented results which indicate that the phenomena of exceptionally hard problems in the easy region of problem spaces is mainly an algorithm dependent one. This is in contrast to the phase transistion peak in median search cost, which is caused by problems that appear to be hard for all algorithms. We have also found for complete search algorithms that by using backjumping (in this case conflict-directed backjumping) and sophisticated variable selection heuristics we can significantly reduce the frequency of exceptionally hard problems for an algorithm.

The best results for soluble problems were obtained by the incomplete, stochastic search algorithm GENET. In fact we did not find any exceptionally hard, soluble problems for this algorithm. Although using stochastic search would mean giving up completeness, for large problems this may not be too much of a sacrifice, since a complete search may take too long to perform in practice if a problem is insoluble. Whether one should always use stochastic search in the easy region is another issue. Our results suggest that for smaller problems the best of the complete algorithms we looked at is quite acceptable, and only for larger problems should one consider using in-

complete, stochastic search.

References

[1] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[2] P. Cheeseman, B. Kanefsky, and W. Taylor. Where the really hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, volume 1, pages 331–337, 1991.

[3] A. J. Davenport, E. P. K. Tsang, C. J. Wang, and K. Zhu. GENET: A connectionist architecture for solving constraint satisfaction problems by iterative improvement. In *Proc, 12th National Conference on Artificial Intelligence*, volume 1, pages 325–330, 1994.

[4] I.P. Gent and T. Walsh. Easy problems are sometimes hard. *Artificial Intelligence*, 70:335–345, 1994.

[5] I.P. Gent and T. Walsh. The hardest random SAT problems. In B. Nebel and L. Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence. 18th German Annual Conference on Artificial Intelligence*, pages 355–366. Springer-Verlag, 1994.

[6] M. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46, 1993.

[7] T. Hogg and C.P. Williams. The hardest constraint problems: A double phase transistion. *Artificial Intelligence*, 69:359–377, 1994.

[8] S. Minton, M. Johnston, A.B. Philips, and P. Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

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

[10] P. Prosser. Exceptionally hard problems. USENET Article 3473dh$27c@@todd-06.cs.strath.ac.uk, comp.constraints, September 1994.

[11] B. M. Smith. In search of exceptionally difficult constraint satisfaction problems. In *11th European Conference on Artificial Intelligence, Workshop on Constraint Processing*, pages 79–86, 1994.

[12] E.P.K. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1993.

[13] J. S. Turner. Almost all k-colorable graphs are easy to color. *Journal of Algorithms*, 9:63–82, 1988.

Appendix—results

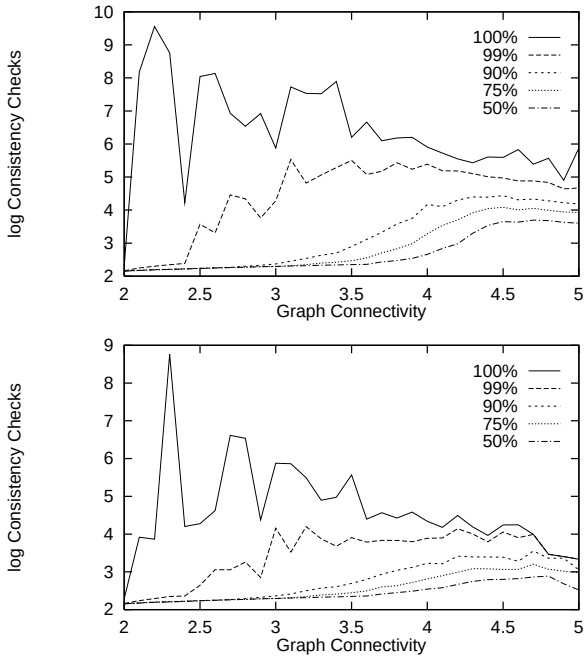

Figure 3: fc-cbj-ff on 50-node graph 3-colorability problems

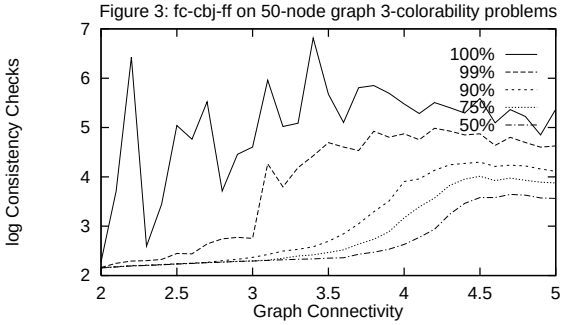

Figure 4: fc-cbj-ff on soluble 50-node graph 3-colorability problems

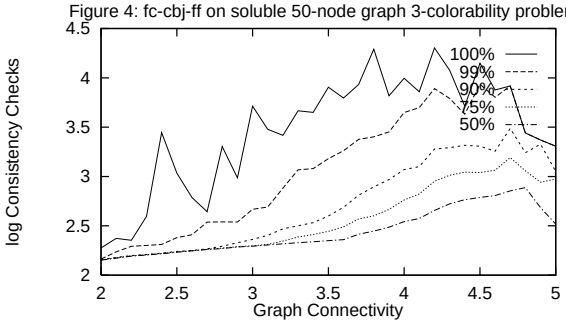

Figure 5:fc-bz on 50 node graph 3-colorability problems

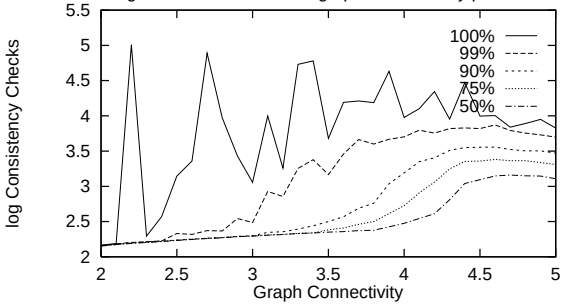

Figure 6:fc-bz on soluble 50 node graph 3-colorability problems

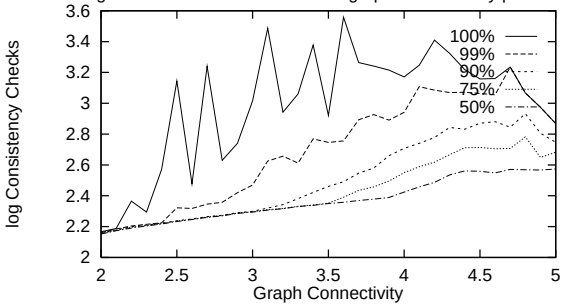

Figure 7:fc-cbj-bz on 50 node graph 3-colorability problems

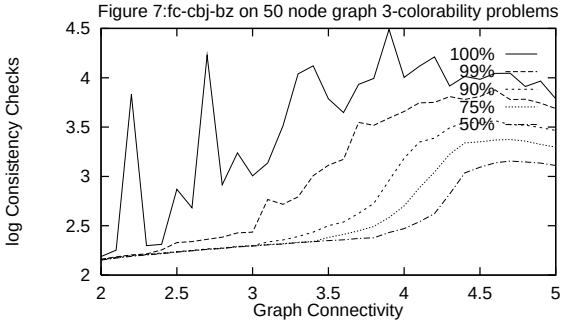

Figure 8:fc-cbj-bz on soluble 50 node graph 3-colorability problems

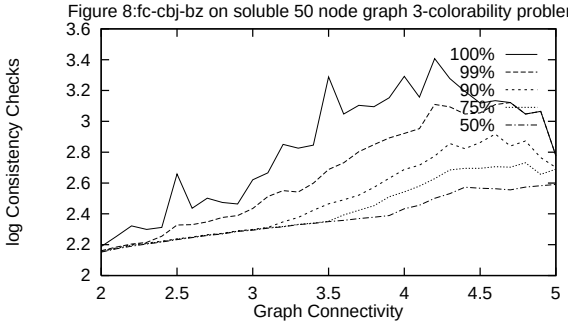

Figure 9: genet on 50-node graph 3-colourability problems

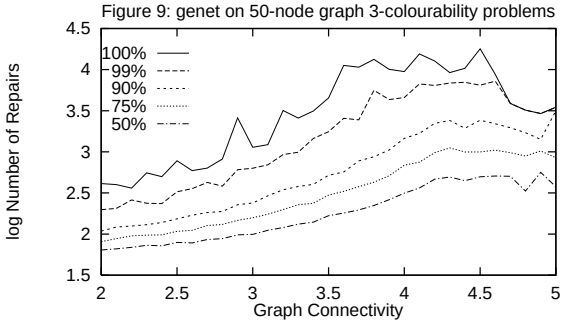

Figure 10: fc-cbj-ff on 100-node graph 3-colorability problems

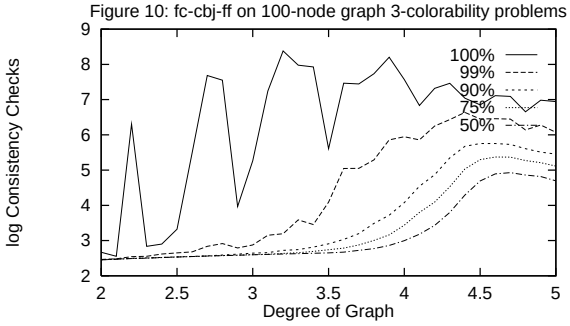

Figure 11:fc-cbj-ff on soluble 100 node graph 3-colorability problems

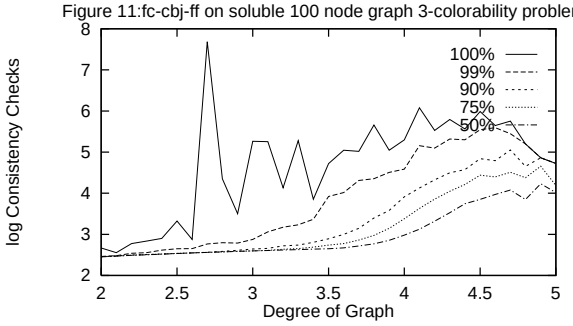

Figure 12: fc-bz on 100-node graph 3-colourability problems

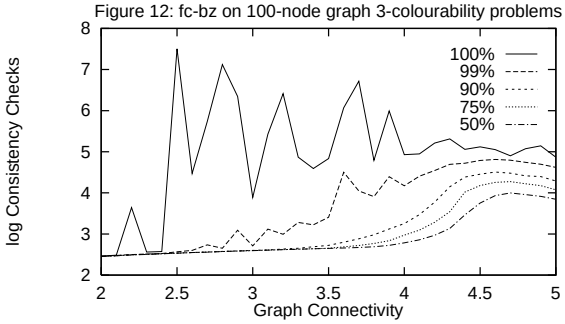

Figure 13: fc-bz on soluble 100-node graph 3-colourability problems

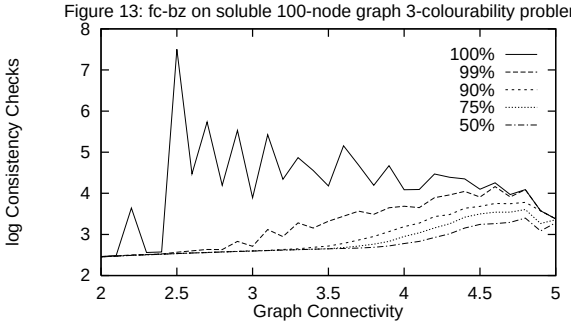

Figure 14: fc-cbj-bz on 100-node graph 3-colourability problems

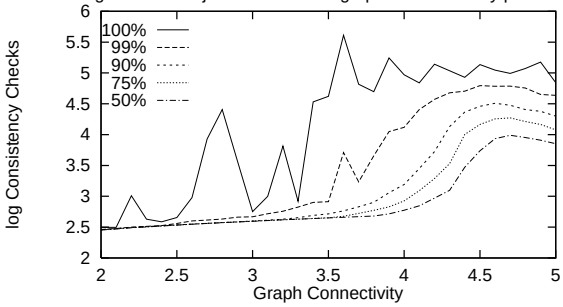

Figure 15: fc-cbj-bz on soluble 100-node graph 3-colourability problems

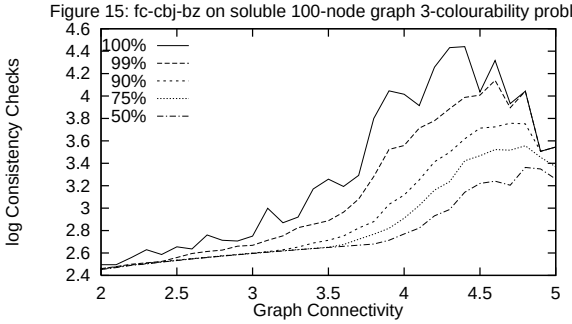

Figure 16: genet on 100-node graph 3-colourability problems

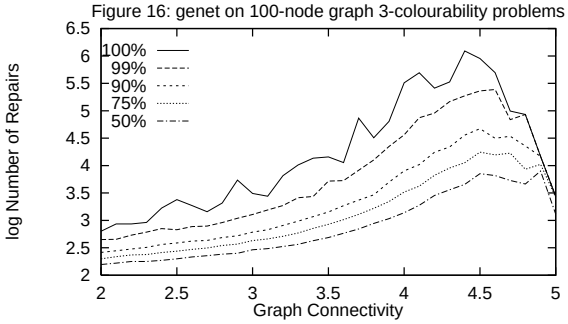

Figure 17: fc-cbj-bz on 200-node graph 3-colorability problems

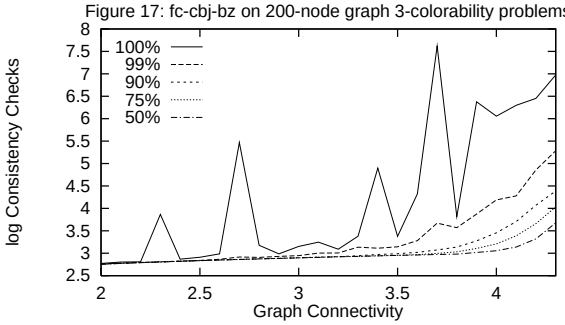

Figure 18: fc-cbj-bz on soluble 200-node graph 3-colorability problems

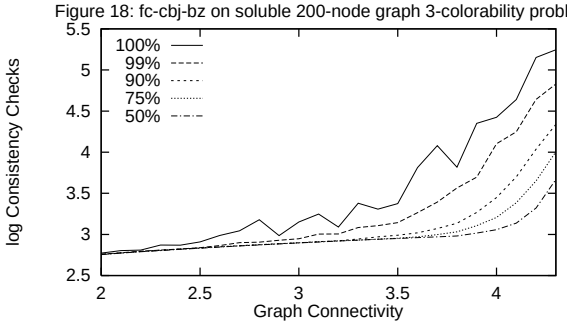

Figure 19: genet on 200-node graph 3-colorability problems

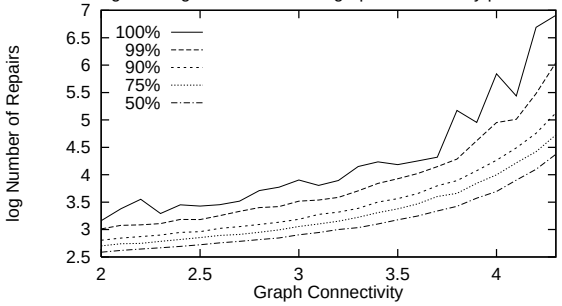

Figure 20: fc-cbj-bz on 400-node graph 3-colorability problems

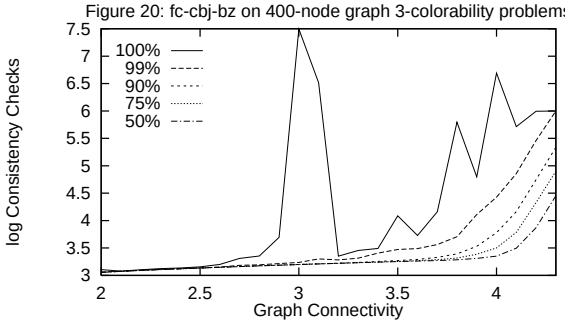

Figure 21: fc-cbj-bz on soluble 400-node graph 3-colorability problems

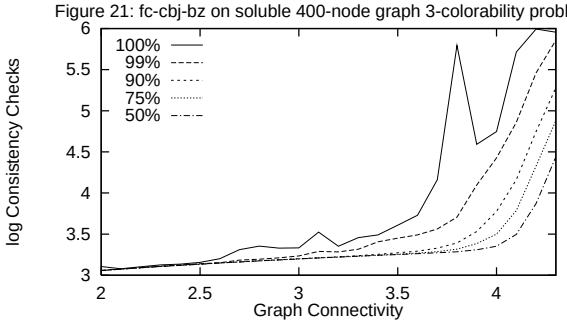

Figure 22: genet on 400-node graph 3-colorability problems