Modelling More Realistic SAT Problems *

Andrew Slater

Computer Sciences Laboratory, Australian National University, 0200, Canberra
andrew.slater@cslab.anu.edu.au

Abstract. The satisfiability problem is widely used in research on combinatorial search and for industrial applications such as verification and planning. Real world search problem benchmarks are not plentiful, yet understanding search algorithm behaviour in the real world domain is highly important. This work justifies and investigates a randomised satisfiability problem model with modular properties akin to those observed in real world search problem domains. The proposed problem model provides a reliable benchmark which highlights pitfalls and advantages with various satisfiability search algorithms.

1 Introduction

The truth value of a formula of classical propositional logic is totally determined by the truth assignments to the propositional variables from which it is built. A formula is satisfiable if at least one truth assignment to its propositional variables makes the whole formula true. Given a formula using a set of propositional variables of size n, deciding whether the formula is satisfiable is known to be NP-complete [1]. For this reason propositional satisfiability is a prototypical search problem commonly used to analyse search behaviour and problem hardness in combinatorial search. The advanced state of research in satisfiability has made it attractive for many real world problem domains such as planning and verification. Real world problems all have some kind of structure; they differ from random problems in that they have non-uniform distributions and that certain parts of the problem (or inferences from that part) are integral units to the problem as a whole. Real world problems model various interactions with real world objects. Consider the problem of verification of hardware or software, where we would at least expect a modular structure in terms of subroutines or logical integrated circuit components.

Our investigation focuses on identifying generic properties of real world search problems. We are motivated by the desire for a better understanding of search and search problems in the real world, and the limited availability of real world benchmarks. It is evident that, on the whole, real world problems do have exploitable properties which are independent of any particular classification of

* Appears in: Proceedings of the *15th Australian Joint Conference on Artificial Intelligence*, LNAI 2557 pp 591-602, Springer 2002.

problem that we want to solve. This is shown by the performance of particular advanced search techniques that are successful on sets of real world benchmarks[2]. This work investigates modelling modular structure to emulate real world problem structure. Section 2 reviews other approaches for generating realistic satisfiability search problems. Section 3 investigates the justification for using modelling modularity in pseudo-real search problems. Our proposed problem model is presented in Section 4, and experimental analysis appears in Section 5 and 6. Section 7 discusses our observations with respect to related work.

2 Structured Problem Generators

There are two obvious approaches to modelling real world structure in problems. The first is to extend the description of a real world problem domain in order to gain further generalisation, but hopefully retain the structural properties of that problem type. The second is to attempt to construct a parameterised problem model by introducing some structural constraints in the problem definition.

There are many approaches based on generating problems using a given problem domain structure. Quasigroup completion problems encoded as constraint satisfaction were proposed by Gomes and Selman [3]; a partial definition of a quasigroup is provided, and must be completed to be solved. Generating satisfiable problem instances within the quasigroup domain is further investigated in [4]. Other approaches include problem generation by encoding the parity problem (see [5]) and the domain of cryptography [6]. A more general approach called *morphing* was devised by Gent *et al.* [7]. Their work categorises some approaches of "morphing", each of which defines a translation from given problem definitions to a new problem via the introduction of noise. The morphing technique can be a very powerful tool for generating a variety of similar problem instances.

The alternative approach is to define a model that incorporates some structural content in a problem and allows the problems to be randomly generated based upon that model. This is the approach taken in this work. We have argued that modularity is a good generic structural property of many real world domains and model this by combining instances of random 3-SAT problems as modules or *clusters*. Our model will be discussed in detail in Section 4. There are similar approaches which are briefly reviewed here.

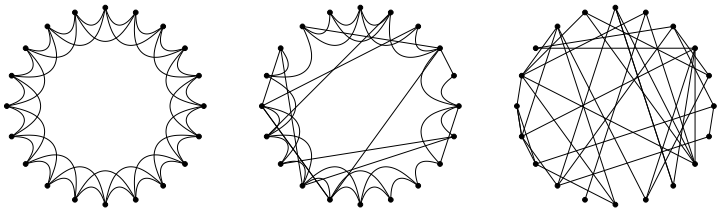
Two methods for random generation of problems with possible real world similarity were proposed by Rish and Dechter [8]. The first model is similar to our model in that its modular components are random 3-SAT problems, which they call sub-theories. The main difference is in [8] the sub-theories are connected in a chain by joining neighbouring clusters using a single binary clause. Their observations are discussed in Section 7. Their second model, the $(k, m) - tree$, generates a randomly conglomerated set of "cliques" for which a number of clauses is defined. This model has a graph theory analogy but the details and implications for satisfiability problems are unclear. Finally we note that there are other problem generators based around pattern construction and repetition for other related problem domains (e.g. graph colouring, network topology)[9–11].

3 Justifying Modularity

The observation of modular structure in search problems has been investigated by Walsh [12], motivated by the observations of "small world" phenomena observed by Watts and Strogatz [11]. The "small world" model describes a relational system which is somewhere between a completely structured object and a completely random object. Several small groups are highly related, but relations between some members of different groups also exist. Our work finds justification in Walsh's observations of small world phenomena in search problems.

In order to capture the notion of the small world topology, Watts and Strogatz combine quantified structural properties of a graph. A graph topology may be characterised by computing the *characteristic path length* and *the clustering coefficient*. The characteristic path length L specifies the average shortest distance between any two vertices in the graph. The clustering coefficient C is defined as the average fraction of the fully connected graph that any given vertex may be a member of. C may be computed as the average C_v for all vertices v, where C_v is defined as follows. A vertex v with k_v neighbours can have have at most $k_v(k_v - 1)/2$ edges between them (the fully connected graph K_{k_v}). C_v is the ratio of the actual number of edges between any of v or its neighbours to the total number of possible edges. These parameters are used to look at a range of graphs from the completely regular, or structured, to the completely random.

A graph defining a ring lattice, where each vertex is joined to k nearest neighbours, has a high clustering coefficient. We construct a random graph of the same size by randomly assigning the edge connections. Random graphs tend to have smaller characteristic path lengths and much smaller clustering coefficients relative to their regular counterparts. With appropriate bounds guaranteeing graph connectivity these two topologies are used to find the small world phenomena.



(a) Ring Lattice $p = 0$ (b) Small World $0 < p < 1$ (c) Completely Rewired $p = 1$

Fig. 1. Examples of "connectedness" with a graph of 20 vertices: (a) shows a ring lattice where each vertex is connected to the four closest neighbours, (b) shows a partially rewired ring lattice where each edge is rewired with probability p, (c) shows the graph where every edge has been rewired.

	L	L_{rand}	C	C_{rand}	μ
Film Actors	3.65	2.99	0.79	0.00027	2396
Power Grid	18.7	12.4	0.08	0.005	10.61
C. elegans	2.25	2.25	0.28	0.05	4.755

Table 1. Analysis of real networks: Film Actors – from a database of collaborations between actors in feature films; Power Grid – the electrical power grid for the western United States; *C. elegans* – the neural network from a nematode worm. The values shown are the characteristic path length, the characteristic path length for a random graph of the same size, the clustering coefficient, the clustering coefficient for a random graph of the same size, and the proximity ratio μ (see text). Note that $L \gtrsim L_{rand}$ and $C \gg C_{rand}$ which also corresponds to the small world characterisation from the Watts and Strogatz rewiring experiments. From [Walsh, 1999].

A small world graph will have a relatively high clustering coefficient and a small characteristic path length. Watts and Strogatz use a "rewiring" concept, whereby an edge is rewired to a new destination with probability p. The details of the rewiring process are less important – different approaches appear in [7]. At $p = 0$ the graph is completely regular, and at $p = 1$ the graph is completely random. Watts and Strogatz find small world graphs do exist in the interval $[0, 1]$ for p. Their analysis shows a small world network is one where $L \gtrsim L_{rand}$ and $C \gg C_{rand}$, where C_{rand} and L_{rand} are the characteristic path length and clustering coefficient from a random graph with the same number of vertices and edges. Figure 1 shows an example of the ring lattice, the partially rewired small world graph, and the random graph for visual comparison.

Watts and Strogatz apply the small world topological analysis to some examples of networked relationships and find that the small world network characterisation according to their model is apparent. Table 1 shows values for their results. Walsh extends the small world analysis of Watts and Strogatz by including the *proximity ratio*. It is defined as a normalised relationship between the characteristic path length and the clustering coefficient of a graph:

$$\mu = \frac{C}{L}\frac{L_{rand}}{C_{rand}}$$

where C_{rand} and L_{rand} are the characteristic path length and clustering coefficient from a random graph with the same number of vertices and edges. This measure allows the "small world" characterisation of one graph to be compared against another. Note that a random graph will have a proximity ratio of 1 and lattices will have small proximity ratios. For small world graphs $\mu \gg 1$.

Walsh demonstrates that several existing benchmark problems for graph colouring have high proximity ratios. Furthermore he shows that high proximity ratios can be calculated for timetabling benchmarks and quasigroup problems. In a search cost study for graph colouring Walsh also shows topological features in search problems can have a large impact on search cost [12].

Graph colouring problems have an obvious translation for network topology analysis. In the case of satisfiability, other notions defining the relations between variables, such as co-occurrence in clauses, can be used to determine the proximity ratio, and studies show high proximity ratios for a topological interpretation of some satisfiability problem benchmarks [7]. The concept of the clustering coefficient requires that the modularity be exposed at an atomic level. This means that a sub-problem or module is only identifiable when the clustering coefficient in a sub-problem is high. For variable co-occurrence this means that a "cluster" would be very small, and in practice we should not expect this. Further development of small world topological analysis may yield better formal characterisations for satisfiability problems, however the studies of small world properties in real world search problems produces a strong argument that modular structure exists and appears to be a strong feature of real world problems.

4 A Problem Model for Clustering

In order to capture modularity in a parameterisable problem set we propose a problem model based on the random generation of fixed size modules or clusters. The model uses a set of individual random 3-SAT problems which are "connected" by a small set of extra clauses.

Formally we define the random clustered problem model as follows. A clustered problem instance has n variables, m clauses, c clusters and p percent links. We may use the clause to variable ratio r, a common parameterisation in random 3-SAT experiments to indicate the 'hardness' of each cluster – each cluster has n/c variables, and $(1 - p/100)m$ clauses (ignoring remainders). As p increases, the structure of the individual clusters decay, and eventually the problem will become a regular random 3-SAT problem. To generate the problems each cluster is generated as a separate random 3-SAT problem. The remainders of clauses and variables can be distributed as evenly as possible e.g. cluster i $(0 \leq i < c)$ has n/c variables $+1$ if $i < (n \textbf{ mod } c)$. The fraction of clauses that act as links are randomly generated using the entire set of variables. We wish to investigate the effects of structural changes such as this. Furthermore we want to define some level of difficulty within the individual clusters so that they appear to be a separate sub-problem to the search algorithm. We also want the link clauses to make the problem appear in some sense as a whole.

The aim of this problem model is to capture structure through some arbitrary modularity. It does not necessarily have the quantifiable properties of a "small world" problem, since the size of each individual cluster can be varied. A random cluster problem will probably only have a small world characterisation when the cluster sizes are very small since the measurement of the proximity ratio requires relationships at the atomic level. For small clusters, the clustering coefficient will be larger, and overall the characteristic path length should be shorter between most variables. On a meta-level, however, we emulate the small world situation by modelling the local interactions by clusters, and the global interactions by the links. This small world concept is relatively independent of the cluster size.

We shouldn't expect the problems to be particularly difficult if the cluster problems are not difficult. The problems should measure the ability of a search algorithm to "concentrate" on a particular problem, or at least identify which parts of the search space are relevant to the current search state. Consider why some large real world problems are not as difficult as a hard random 3-SAT problem. A good search algorithm will be able to identify, through some means, an appropriate ordering of the search so that each cluster is solved individually – a divide and conquer style approach.

5 Experimenting with the Model

Experiments were performed to measure search cost for various parameterisations of the proposed random cluster model. The experimentation reveals that the theory behind the model can predict search algorithm behaviour. There were some interesting exceptions in initial experiments which are discussed below.

The satisfiability search system `satz` [13] was chosen for performing preliminary experiments. It uses a Davis-Putnam-Logemann-Loveland (**DPLL**)[14] based search algorithm with a powerful choice heuristic. It is an efficient and reliable implementation, and furthermore is very capable in solving large random 3-SAT problems. This allows testing on larger problem sizes and the use of large sample sets. It is posited that the use of a state of the art system is a reasonable approach in testing the proposed benchmark problems. Additionally, the use of large cluster sizes allows greater potential for "isomorphic richness" in individual clusters. Very small random 3-SAT clusters is not representative. Experiments were done using a Sun UltraSPARC II (248MHz) processor.

To generate the random clustered problems, a problem generator that takes the problem set parameters and produces the set of problems was implemented. Clauses which contain a duplicate variable are not included, and problems that are unconnected are not produced; i.e. if the set of links does not join each cluster through co-occurrence then the links are regenerated.

Clustered problem sets for a range of number of clusters, and a variety of different percentage links, were generated for initial experiments. Each problem had 200 variables and 840 clauses. Surprisingly, at around 5 clusters the problems became significantly harder to solve. For low values of percentage links, a number of the problems could not be solved within the 5 minute CPU time limit set for each problem. This contrasts severely with the CPU time taken to solve a single cluster problem (the regular random 3-SAT problem) of the same size. For a set of single cluster problems a median time of 0.56 seconds was measured, where 69 percent of the instances were satisfiable. Figure 2 shows the results computed for the cluster problems using 15 percent of the clauses as links. The drop in satisfiability is consistent with predictions, but the performances are not. As the percentage of links increases, the level of difficulty drops. The most difficult problems are the ones with 5, 6 or 7 clusters. Increasing the percentage links values causes the difficulty to subside.

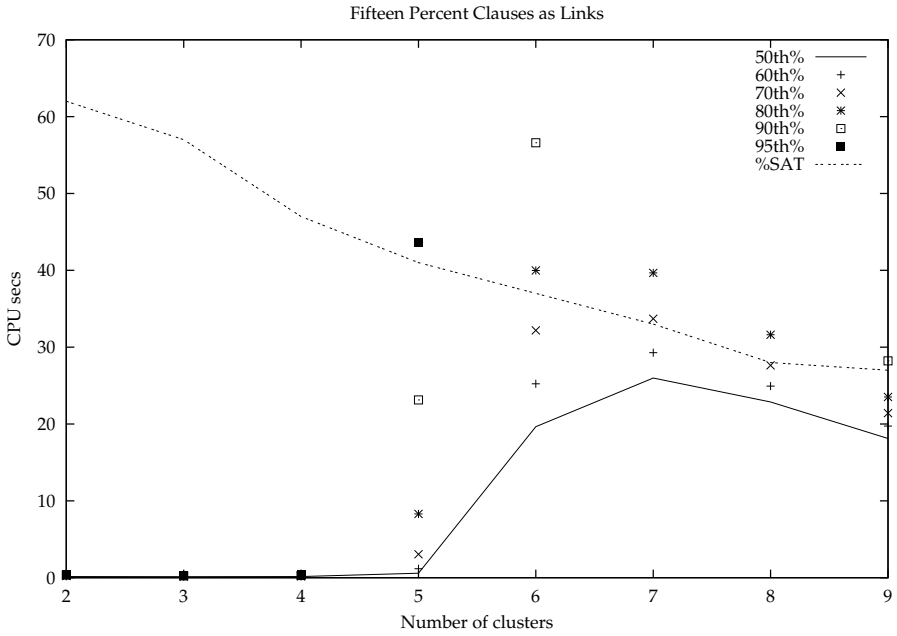


Fig. 2. Time taken for clustered problems of 200 variables and 840 clauses. The number of clusters ranges from 2 to 9. The number of clauses that are used as links is 15 percent of the total number of clauses. 200 problems were tested for each cluster size. The median values are shown by the solid curve, and various percentile points are plotted. The percentage of problems that are satisfiable is also plotted as a dashed curve. Note that for a single cluster the time taken is less than 1 second.

It appears that in the initial experiments the difficulty is a function of the percentage of links as well as the number of clusters. The source of the difficulty is not within the **DPLL** search part of `satz`, but in a resolution based preprocessing technique it uses. It is claimed by the authors of `satz` that this technique reduces the search cost by about 10 percent for hard random 3-SAT, and a variety of speed-ups is seen for benchmark problem classes [15]. The fine details of this procedure are not important - the key issue is that a weakly controlled resolution process occurs before any **DPLL** style search is performed. Although resolvent sizes are limited this does not bound the process strongly enough to cope with the situations presented in some of the random cluster problems. For the apparently difficult random cluster problems generated, `satz` created far too many resolvents for the preprocessing technique to cope with. The probability of co-occurrence of variables is high for small random problems, thus the potential number of small resolvents in each cluster will also be high. The small world qualities of these cluster problems is likely to be higher merely due to their size, and it is the "cliqueishness" of each cluster that causes the breakdown of the preprocessing technique. It is highly likely that a fuller version of resolution could

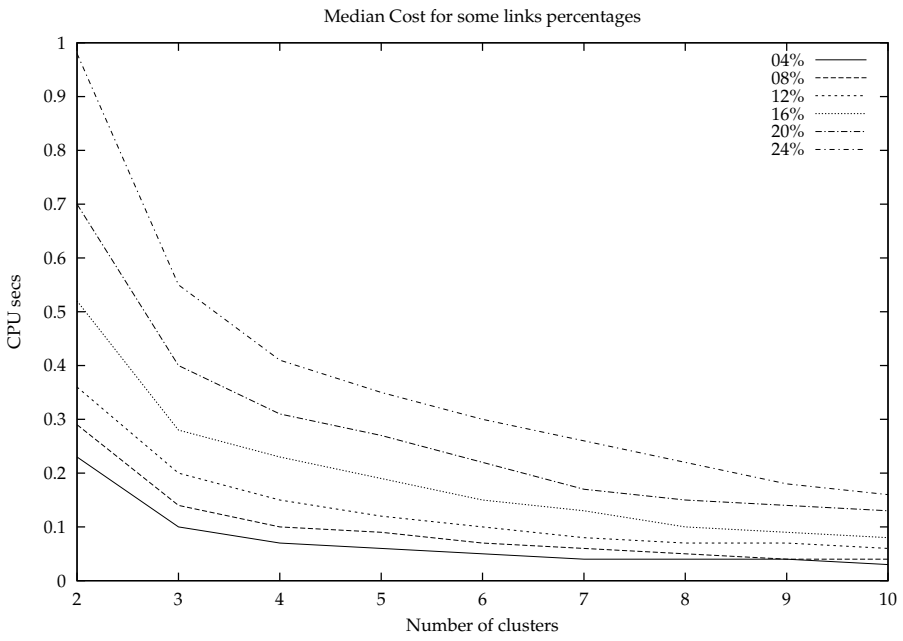


Fig. 3. Median time taken for clustered problems of 250 variables and 1061 clauses (a global clause to variable ratio of about 4.24) for a variety of percentage links. The key indicates the percentage of the clauses that are used as links for particular curves. The number of clusters ranges from 2 to 10. 1000 problems were used for each cluster size. Note that for a single cluster problem (a random 3-SAT problem of 250 variables and 1061 clauses) the median time taken for 1000 problems is around 4.2 seconds.

actually solve the individual cluster problems as in [8], which shall be discussed in further detail in Section 7. A further set of experiments was performed with the `satz` resolution actions disabled on problems with 250 variables and 1061 clauses, using from 2 to 10 clusters, for a variety of percentage links values. For these experiments 1000 problems were tested for each combination of parameters used. Figure 3 shows the median search cost in CPU time for a variety of percentage links, against the number of clusters in the problems. As expected, the cost decreases as the number of clusters increase since the individual sub-problems are easier to solve. For higher values of percentage links the problems are more like the single cluster, or basic random 3-SAT, and are harder to solve than the problems with well defined clustering. This may be because there is less information about where the search algorithm should concentrate as more links will direct the branching heuristics to a different sub-problem. Also note that the percentage of satisfiable problems decreases with cluster size. This is due to the compounding of the individual probability of any cluster being unsatisfi-

able. A problem with a higher number of clusters is easier when an unsatisfiable sub-problem is identified early, as search is terminated.

6 Measuring A System's "Concentration"

Earlier it was hypothesised that the random cluster model could be used to measure how well a satisfiability search system could "concentrate" on a problem. By this we mean how well it can search and solve a sub-problem before moving onto some other part of the problem. A variety of state of the art systems were tested with the random cluster model. In order to get an idea of how well they coped, their ability was compared to the basic random 3-SAT problems by ranking the performance results for both problem types. Those that rank highly in the random cluster problems should have a good ability to "concentrate". By comparing the rankings to those for the random 3-SAT problems we should see the relative difference when modularity is introduced. The systems used were:

`satz` [15] The initial system used for our experimentation which relies solely on its ability to make the best choices in the search tree. This system is renowned for its performance on random 3-SAT problems, so the individual cluster problems should not be difficult for it.

`posit` [16] A somewhat earlier system which uses a combination of methods to select good variable choices and prune search space. Freeman's thesis demonstrates that it is successful in solving a wide variety of benchmark problems as well as random SAT problems [16].

`relsat` [2] An advanced system that utilises a version of dynamic backtracking and learning techniques. Its ability to reason about relevancy in search space should make it an ideal candidate for modular problems, and it is generally very successful on real world benchmark problems.

`OKSolver` [17] A system which uses a variety of pruning techniques and branch ordering heuristics which seems to be motivated by complexity analysis.

`sato` [18] This system utilises an alternative data structure for search reasoning and has proven very successful on quasi-group problems, having solved open problems in this domain.

The results are ranked in order of smallest median score. Both sets of problems contained 1000 samples. The random 3-SAT problems had 200 variables and a clause to variable ratio of 4.25. The random cluster problems had 450 variables and a clause to variable ratio of 4.25 with 9 clusters and 20 percent of the clauses acting as links. We stress that the two sets of results are produced to gain a ranking only, as we cannot directly compare the performances on these potentially very different problem sets. Table 2 a) shows statistical data gathered from the random 3-SAT experiment. Both `satz` and `posit` outperform the other systems, and the other median scores are several times greater than the two top systems. The results shown in Table 2 b) show quite a different trend. Notably `relsat` copes far better with the random cluster problems. The bottom three systems again have median costs several times greater than the best, with `sato` and `posit` being slower by more than an order of magnitude. The rankings of a selection of different systems show that the search method is important

System	Average	Median	Maximum	Minimum	Std Dev.
`posit`	0.345	0.34	1.29	0.03	0.198
`satz`	0.355	0.36	0.99	0.04	0.195
`OKSolver`	1.123	1.13	4.54	0.05	0.819
`relsat`	1.792	1.69	7.72	0.07	1.360
`sato`	3.891	3.02	28.85	0.00	3.745

a) Ranked Results for One Cluster Problems

System	Average	Median	Maximum	Minimum	Std Dev.
`satz`	10.806	5.68	151.57	0.19	15.140
`relsat`	13.051	8.56	137.30	0.29	14.452
`OKSolver`	28.754	19.58	277.38	0.18	31.722
`sato`	167.571	58.30	3852.26	0.08	333.333
`posit`	151.737	65.69	3392.30	0.19	266.545

b) Ranked Results for Nine Clusters Problems

Table 2. These tables show an ordering of run-time statistics for a selection of state-of-the-art systems. Each table is ordered by increasing median value. Note that the ordering of the other statistics is similar but does not always agree. Table a) shows results and ranking for regular random 3-SAT problems. Table b) shows results and ranking for the clustered problems. The difference in the ordering of the two tables indicates how the different methods cope with the random cluster model.

for solving this kind of modular structure. Since the individual problems are random 3-SAT problems, we would normally expect that a system that solves these well will cope with random clusters. This was not the case with `posit` whose search technique was not powerful enough to detect the sub-problems as well as the best ranked systems. `relsat`, known for an ability to deal with real world domain problems fared much better in the rankings for the random cluster model problems than with the random 3-SAT problem. This is evidence that the modular structure is apparent to its search method, and furthermore suggests that the random cluster model is a far better model of the real world domain than the random 3-SAT model.

7 Related Work

We previously noted the work of Rish and Dechter [8] who independently created a structured problem model they call random 3-CNF chains. In a series of experiments that appears to show Directional Resolution (DR) is more capable than DPLL algorithms, they find anomalously hard problems for their implementation of DPLL. The search times differ by orders of magnitude. We found no anomalous hardness spikes in any of our experiments using DPLL style algorithms, bar the resolution based preprocessing problems. The difference in observations is

most likely to be due to the fact that our clusters are not chained – the path length from any cluster to another is very short. Rish and Dechter pose that when a sub-problem at the end of the chain is unsatisfiable then the entire problem must be re-solved. This would infer that their algorithm chose a particularly unfortunate ordering, and it seems that such a situation would be reasonably rare. In our experiments we sampled tens of thousands of large problems with different parameterisations. It is conceivable that the same unfortunate ordering could occur when the last cluster to be solved was unsatisfiable. However we did not notice any problems that were as extremely hard as those observed in [8]. We suspect that a combination of the heuristics used and the scale of the model used by Rish and Dechter is also responsible for large search costs. In fact, for the number of variables in the problems, some of the worst search costs suggest that the supposedly advanced choice ordering has failed badly. Since this ordering is based on `tableaux` [19], we suspect that the empirically derived choice mechanism is over-fitted to random 3-SAT problems and will prefer to work on binary links rather than sub-problems. We also believe that the size of the sub-problems Rish and Dechter use is a significant factor. The probability of co-occurrence for any pair of variables is extremely high. In this situation a resolution based approach is bound for success. Larger sized chain components will eliminate this advantage. By adding back-jumping to their DPLL implementation Rish and Dechter are able to avoid most difficult problems, but they do suggest a possible phase transition phenomenon in chain problems. This seems false in view of the other experiments.

8 Conclusions

Modularity has been identified as an important structural concept based on intuitive reasoning and the work of Walsh [12]. The proposed problem model based on clusters of random 3-SAT problems gives a simple framework to generate modular problems that can be manipulated via the parameter set that defines them. This problem model generally behaves as the theory predicts when used in experimentation. However, our experimentation revealed possible problems with search methods that integrate resolution style techniques. This is because the nature of small connected sub-problems can yield an exponential explosion in the number of resolvents. The success of Rish and Dechter's DR algorithm indicates that more advanced implementations of resolution algorithms to perform the search are far less likely to succumb to the problems we observed. Our observations reveal further explanations for the performance results observed in [8] for a related problem model and highlight the fact that choice heuristics derived from 3-SAT experimentation may not translate to other problem domains.

A reasonable search algorithm is one which is able to "concentrate" on a problem by tackling sub-problems. We observed this behaviour with some state of the art systems using the random cluster model. These results indicate that the proposed model has much better real world similarity than basic random models. That problem difficulty is likely to be only as hard as the contained

sub-problems is reinforced by recent work in modularity detection [20]. The proposed model yields a simple framework for generating large sets of test cases with real world properties for satisfiability search algorithms.

References

1. S.A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, 1971.
2. Roberto J. Bayardo, Jr. and Robert C. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *AAAI-97*, 1997.
3. C. Gomes and B. Selman. Problem structure in presence of perturbations. In *AAAI-97*, 1997.
4. D. Achlioptas, C. Gomes, H. Kautz, and B. Selman. Generating satisfiable problem instances. In *AAAI-2000*, 2000.
5. David S. Johnson and Michael A. Trick, editors. *Cliques, Coloring, and Satisfiability: the Second DIMACS Implementation Challenge*, volume 26. American Mathematical Society, 1993.
6. F. Massacci. Using walk-sat and rel-sat for cryptographic key search. In *IJCAI-99*, 1999.
7. I.P Gent, H.H. Hoos, P. Prosser, and T. Walsh. Morphing: Combining structure and randomness. In *AAAI-99*, 1999.
8. Irina Rish and Rina Dechter. Resolution versus Search: Two strategies for SAT. *Journal of Automated Reasoning*, 25(1-2):225–275, 2000.
9. Tadd Hogg. Refining the phase transition in combinatorial search. *Artificial Intelligence*, 81, 1996.
10. S. Grant and B.M. Smith. Where the exceptionally hard problems are. In *Proceedings of the CP-95 workshop on Really Hard Problems*, 1995.
11. D.J. Watts and S.H. Strogatz. Collective dynamics of 'small world' networks. *Nature*, 393:440–442, 1998.
12. Toby Walsh. Search in a small world. In *IJCAI-99*, 1999.
13. Chu Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *AAAI-97*, 1997.
14. Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, July 1962.
15. Chu Min Li and Anbulagan. Look-ahead versus look-back for satisfiability problems. In *CP-97*, 1997.
16. Jon W. Freeman. *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, University of Pennsylvania, 1995.
17. Oliver Kullman. Heuristics for sat algorithms: Searching for some foundations. Technical report, Dept. Computer Science, University of Toronto, 1999.
18. H. Zhang. SATO: An efficient propositional prover. In *CADE-14*, 1997.
19. J. M. Crawford and L. D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81:31–57, 1996.
20. E. Amir and S. McIlraith. Partition-based logical reasoning. In *KR-2000*, 2000.