Local Search on Random 2+p-SAT

Josh Singer[1], **Ian P. Gent**[2] and **Alan Smaill**[1]

Abstract. Random 2+p-SAT interpolates between the polynomial-time problem Random 2-SAT when $p = 0$ and the NP-complete problem Random 3-SAT when $p = 1$. At some value $p = p_0 \approx 0.41$, a dramatic change in the structural nature of instances is predicted by statistical mechanics methods. This is reflected by a change in the typical cost scaling for a complete search method TABLEAU, seen experimentally. We show empirically the same change of of behaviour in the local search algorithm NOVELTY$^+$, a recent variant of WSAT. Between $p = 0.3$ and $p = 0.5$ we see typical cost scaling of NOVELTY$^+$ at the 50% satisfiability point apparently change from slow polynomial growth to superpolynomial. That this behaviour is seen in two such different algorithms lends credibility to the hypothesis that there is change of typical-case complexity around p_0.

Previous work linked the emergence of a *backbone* of fully constrained variables to the cost peak seen in Random k-SAT. Initial experiments suggest that for those instances whose cost was typical, backbone size is no larger for $p = 0.5$ than for $p = 0.3$, implying that this property is not wholly responsible for any typical cost scaling change. A preliminary study shows that the backbones of instances at $p = 0.5$ are more sensitive to the removal of clauses than those at $p = 0.3$. This "backbone fragility", which has previously been linked to local search cost, may cause the drastic increase.

1 INTRODUCTION

The theory of NP-completeness is one of worst-case analysis. Establishing a decision problem to be NP-complete guarantees that for any algorithm, some instances of the problem will require an exponential amount of effort to solve, unless P=NP. However, a probabilistic distribution of an NP-complete problem can be tractable in the average or typical case, as long as the instances requiring exponential effort appear infrequently enough. There are also distributions of NP-complete problems which are widely believed to be intractable (i.e. require superpolynomial cost) in the typical case as well as the worst case since the observed typical cost scales badly for a range of algorithms. Finally, there are tractable special cases of NP-complete problems which have polynomial algorithms.

It is of importance to basic AI research to know *why* some distributions in the average, typical or worst case are intractable whereas others are not. This will help us to identify the core of the search problem which algorithms face and thereby improve them. Hooker [10] suggested that one advantage of an empirical science of algorithms is that it can investigate how performance is related to the characteristics of the problem. In this paper we take an empirical approach to study the emergence of structural characteristics which may be responsible for intractability, with the aim of formulating an explanation of why some distributions may be intractable.

We study the NP-complete satisfiability (SAT) problem. In this paper we use the Random 2+p-SAT model due to Monasson, Zecchina, Selman, Kirkpatrick and Troyansky [15]. By varying p, we may interpolate between Random 2-SAT ($p = 0$), a tractable special case of SAT which has a linear worst case algorithm [2] and Random 3-SAT ($p = 1$). Random 3-SAT is NP-complete in its own right and the "threshold" region of this distribution is conjectured to be intractable in the average and typical as well as the worst case (e.g. Cook and Mitchell [6]).

Monasson *et al.* suggested that the range over which p varies may be divided into two qualitatively different regions delineated by a value p_0 estimated at about 0.41. With $p < p_0$, Random 2+p-SAT is structurally very similar to Random 2-SAT whereas in the region $p > p_0$, Random 2+p-SAT is distinct and appears to share structural properties with Random 3-SAT. Monasson *et al.* also studied the effect of varying p on search cost for TABLEAU, a complete procedure for SAT. To represent typical cost, the scaling of the median was studied. We take the same approach in this paper. Note that a distribution can be tractable in the typical case, but intractable in the average case if difficult instances occur rarely but affect the mean cost. For TABLEAU, the typical cost scaled linearly for $p \leq 0.4$ but, as in Random 3-SAT, it scaled as a simple exponential for $p = 0.6$. If this change in search cost scaling applies to substantially different algorithms and is not peculiar to TABLEAU or its algorithm class, this would suggest more strongly that one or more of the structural properties which emerge at p_0 causes the onset of typical-case intractability. In this paper we analyse the search cost of NOVELTY$^+$ [12, 14] on Random 2+p-SAT. This is a state-of-the-art algorithm from the important class of incomplete stochastic local search procedures. Our experimental data lends credibility to the hypothesis that the typical case complexity changes near p_0.

We also investigate which emergent structural properties cause change in typical cost for local search. The approach we take is to look for structural properties which distinguish typical instances where $p < p_0$ from those where $p > p_0$ and which could conceivably affect the algorithm's operation.

Section 2 gives background information on SAT, Random k-SAT and Random 2+p-SAT and details of the test instances which were used. In Section 3 we discuss the NOVELTY$^+$ algorithm and the experimental conditions under which it was run. Section 4 presents experimental scaling data for the search cost of NOVELTY$^+$ on Random 2+p-SAT. In Section 5 we relate the structural properties of Random 2+p-SAT to the behaviour of local search cost. Finally, Section 6 gives conclusions and suggestions for further work.

[1] Division of Informatics, University of Edinburgh, 80 South Bridge, Edinburgh, EH1 1HN, UK, email: joshuas@dai.ed.ac.uk, A.Smaill@ed.ac.uk

[2] School of Computer Science, University of St. Andrews, North Haugh, St. Andrews, Fife, KY16 9SS, UK, email: ipg@dcs.st-and.ac.uk

2 SATISFIABILITY DISTRIBUTIONS

This section discusses SAT and the distribution and instance collections which were used.

A SAT instance C is a propositional formula in conjunctive normal form. C is a bag of m clauses which represents their conjunction. A *clause* is a disjunction of *literals*, which are Boolean variables or their negations. The variables constitute a set of n symbols V. An *assignment* is a mapping from V to {true, false}. The decision question asks whether there exists an assignment which makes C true under the standard logical interpretation of the connectives. Such an assignment is a *solution* of the instance. If there is a solution, the SAT instance is said to be *satisfiable*.

In k-SAT we specify that each clause must contain exactly k distinct literals. For $k \geq 3$ the problem is NP-complete [5]. Random k-SAT is a parameterised, probabilistic distribution of k-SAT. Each k-clause in a Random k-SAT instance is independently chosen by randomly selecting as its literals k distinct variables from V and independently negating each with probability $\frac{1}{2}$. We do not guarantee that all variables in V are mentioned in C nor that C contains no duplicate clauses. If n is fixed and the ratio m/n is increased, at a critical value $\alpha_{0.5}$ of m/n, Random k-SAT exhibits a threshold, where the probability of satisfiability rapidly changes from 1 to 0. For $k = 2$, the location of $\alpha_{0.5}$ in the limit has been derived analytically, whereas for $k \geq 3$, assuming that in the limit it converges to a constant, it must be estimated (at about 4.25 for $k = 3$) using experiments since to date even advanced analysis can only derive bounds.

Experiments [3, 4, 7, 13] have revealed that average cost to determine satisfiability for both complete and incomplete algorithms of substantially different designs peaks near $\alpha_{0.5}$ in Random 3-SAT. When m/n is less than about 3 in Random 3-SAT, a solution can be found with probability tending to 1, in polynomial time [8]. At $\alpha_{0.5}$ on Random 3-SAT, average cost for complete backtracking algorithms appears to scale as a simple exponential [7, 9]. For incomplete local search algorithms, Gent, Prosser, MacIntyre and Walsh [9] showed that average GSAT cost growth on Random 3-SAT was consistent with a polynomial (at most quartic), where m/n is less than $\alpha_{0.5}$. Parkes and Walser [18] studied larger instances at $\alpha_{0.5}$ and found that WSAT average cost growth was not consistent with any function of the form an^b, but that certain functions which were superpolynomial but which grew more slowly than a simple exponential, did fit the data. The fact that the pattern is invariant across such different algorithms strongly suggests that as m/n is increased towards $\alpha_{0.5}$, the Random 3-SAT distribution acquires a "hardness" property which somehow causes instances to require superpolynomial time to solve.

The value of k affects the worst-case complexity. For $k = 2$ there is a worst-case linear time algorithm and so in Random 2-SAT the hardness property is never acquired. This property must emerge at some point *between* $k = 2$ and $k = 3$; for this reason the Random 2+p-SAT model was developed. A Random 2+p-SAT instance consists of $(1 - p)m$ Random 2-SAT clauses and pm Random 3-SAT clauses. Hence varying p interpolates between 2-SAT and 3-SAT.

Monasson *et al.* [15] show that for $p < p_0$ the Random 2+p-SAT satisfiability probability changes continuously in the limit. Bounds of 2/5 (lower) and 0.695 (upper) have been established for p_0 [1], which Monasson *et al.* estimate numerically at about 0.41. For $p > p_0$ the change in the satisfiability probability is thought to be discontinuous[3].

In this study we used instances generated from the Random 2+p-SAT distribution using various values of p and n. For each combination of p and n we determined the largest value of m for which at least 50 % of the instances were satisfiable. We generated a large set of satisfiable instances at this point, using both TABLEAU [7] and MODOC [21] as SAT testers. Satisfiable instances comprise the *satisfiable phase* of the distribution. We study only the satisfiable phase because local search cannot solve unsatisfiable instances. Table 1 gives details of these test instances.

p	(n, m) points at approx. 50 % satisfiability
0.0	(25,41) (50,74) (100,136) (250,313) (500,598) (1000,1147) (1500,1690) (2000,2228)* (3000,3288)†
0.3	(25,52) (50,95) (100,177) (250,414) (500,799) (1000,1555)* (1500,2299)† (2000,3052)†
0.5	(25,63) (50,115) (100, 217) (250, 520) (500, 1015)* (750, 1511)* (1000, 2005)† (1500, 2991)†
1.0	(25, 112) (50, 218) (100, 428) (150, 640) (200, 853)* (250, 1065)†

Table 1. Each instance collection contained 5,000 instances, except those marked $*$ which contained 2,500 and those marked †, which contained 1,500. Collection sizes were reduced for larger n so that the computation time required was manageable.

3 THE NOVELTY$^+$ ALGORITHM

In this section we discuss the local search algorithm class, the NOVELTY$^+$ procedure and the experimental conditions we used.

Local search procedures try to find a solution to a SAT instance by examining a sequence of assignments. In a run of a local search procedure, the first assignment is usually selected at random. Local search then proceeds by moving from one assignment to another by "flipping" (i.e. inverting) the truth value of a single variable. These flips are generally considered to be the basic unit of search cost. The number of flips taken to find a solution is called the *run length*. Due to the randomness in the algorithm, multiple runs must be made to estimate the cost of an instance for local search. Typically, local search is incomplete for the SAT decision problem: there is no guarantee that if a solution exists, it will be found within any time bound. Unlike complete procedures, most local search algorithms cannot prove for certain that no solution exists. However, the average cost for local search procedures scales much better than that of complete procedures near the threshold region in Random 3-SAT (see e.g. [18]). Therefore local search procedures can have an important role to play in any SAT system where completeness may be sacrificed.

NOVELTY [14] is a variable selection strategy, combining memory and greediness, which was developed for the WSAT local search architecture [19]. In WSAT, only variables appearing in a clause unsatisfied by the current assignment may be flipped. Hoos [12] made a "random walk modification" to the algorithm; the modified algorithm is NOVELTY$^+$. With a small probability *wp* (we followed Hoos in using *wp* = 0.01) the modified algorithm chooses the variable to flip at random from the unsatisfied clause. This modification guarantees that all runs on satisfiable instances will succeed with probability approaching 1 as the run length is increased.

Following Hoos [11], we test the algorithm without a restart mechanism and use the median run length *mrl*(C) as our measure of per instance search cost: this represents a "typical" run of NOVELTY$^+$ on the instance C. We took the *mrl* of 100 runs of the algorithm per

[3] See e.g. Achlioptas, Kirousis, Kranakis and Krizanc [1] for a fuller explanation of of the meaning of (dis)continuous in this context.

instance except on collections marked $*$ where 50 runs per instance were made and collections marked † where 30 runs were made. Following Monasson *et al.* [15] we then look at the typical per-instance cost (i.e. median *mrl*) over the collection as n is increased.

The algorithm also has a *noise* parameter (0.0 $\leq$ *noise* $\leq$ 1.0) which, roughly speaking, represents the probability that the choice of variable to flip is non-greedy. To test the algorithm's performance scaling, this parameter should be optimised if possible. To complicate matters, the optimal *noise* setting at 50 % satisfiability depends on p, n and the measure being optimised. For each instance collection we determined the *noise* setting which was optimal to within 0.05, minimising the total number of flips to perform all runs on all instances to completion[4]. On three collections ($p = 0.3$, $n = 25, 50, 100$) 100 runs per instance were not enough to distinguish a clear *noise* optimum and so 2,000 runs per instance were performed.

On certain collections ($p = 0.5$, $n \geq 500$, $p = 0.3$, $n \geq 1000$) it was not feasible to allow all runs to complete and so instead we imposed a cut-off run length large enough so as not to affect our measurement of median *mrl*. On these collections we used the *noise* setting which minimised the median *mrl* (optimal to within 0.05).

4 NOVELTY$^+$ ON RANDOM 2+p-SAT

In this section we present results of experimental tests of NOVELTY+ on the instance collections described in Section 2.

4.1 $0 \leq p < p_0$: Typical cost appears to be a low polynomial

Some analytic results are known for this class of algorithm when p = 0 (2-SAT). Papadimitriou [16] showed that for a simple random walk algorithm also based on flipping variables in an unsatisfied clause, a run length of $O(n^2)$ is sufficient to solve any 2-SAT instance with probability at least $\frac{1}{2}$. For $p > 0$, Random 2+p-SAT is NP-complete since any 3-SAT instance may constitute the pm 3-SAT clauses.

p	n	Median *mrl* (3 s.f.)	95th perc. *mrl* (3 s.f.)	99th perc. *mrl* (3 s.f.)	Mean run length (3 s.f.)
0	25	15	24.5	30	17.8
	50	30	49.5	61.5	35.1
	100	60	95	115	68.3
	250	147	219	255	161
	500	294	401	463	311
	1000	562	746	862	591
	1500	832	1,060	119	863
	2000	1,090	1,350	1,490	1,120
	3000	1,590	1,950	2,130	1,630
0.3	25	18	31	43	30.6
	50	36	73	138	101
	100	82	189	682	297
	250	208	801	9,310	3,890
	500	464	6,560	54,600	30,400
	1000	979	22,300	142,000	–
	1500	1,580	54,700	481,000	–
	2000	2,290	81,400	$>$ 500,000	–

Table 2. Run length statistics for NOVELTY$^+$ on Random 2+p-SAT for $p = 0, 0.3$

Table 2 gives experimental data for p = 0 and p = 0.3 on Random 2+p-SAT. As well as median *mrl*, we also give 95th and 99th percentiles of *mrl* and the mean run length. The data for the typical cost for $p = 0, 0.3$ is consistent with a dependence on n which is $O(n^b)$ with $b \approx 1$. For $p = 0$ a linear-bounded growth is also observed for the 95th and 99th percentiles and for the mean. This suggests that algorithms of this class may perform well on 2-SAT, or that instances requiring superlinear cost of the algorithms occur rarely in this distribution. For $p = 0.3$ the higher parts of the cost distribution apparently grow more quickly, although these high percentiles were not stable enough to infer a growth function. The more difficult instances appear to become so costly that they affect the growth of the mean run length. The presence of these instances caused large variances in mean run length which meant that more runs per instance were required for the smaller n. Possible ramifications of the superlinear mean are that either the distribution for $0 < p < p_0$ requires superlinear cost in the average case or that NOVELTY$^+$ is a poor choice of algorithm at this level of p if all instances are to be solved.

4.2 $p_0 < p \leq 1$: Typical cost appears superpolynomial

Figure 1 plots the typical (median) *mrl* for the different values of p as n is increased. The data suggests a qualitative change in the scaling nature between $p = 0.3$ and $p = 0.5$. Plotting the growth for $p \geq 0.5$ on a log scale (not shown) the downward curve shows that it is (at least for this range of n) slower than a simple exponential. A log-log scale (Figure 2) reveals a slight upward curve for $p \geq 0.5$ suggesting that the scaling is superpolynomial. This is similar to the experimental average case scaling results for the closely related WSAT/SKC algorithm [18]. Also, in these two important respects scaling for $p = 0.5$ and 1.0 is similar. For $p \leq 0.3$ the typical cost is a straight line, which confirms polynomial dependence on n.

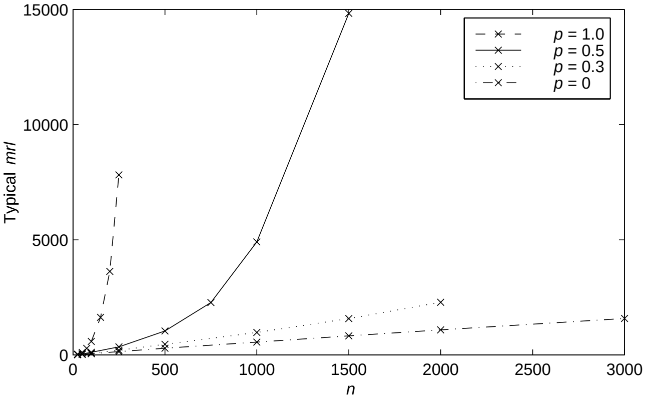

Figure 1. Typical cost

5 STRUCTURE AND COST IN RANDOM 2+p-SAT

The results from Section 4, along with Monasson *et al.*'s work on TABLEAU cost scaling are consistent with the notion that as p is increased, Random 2+p-SAT acquires a property around p_0 which causes the distribution to require superpolynomial computational

[4] So, for example, this is a different optimality criterion from that of McAllester *et al.* [14].

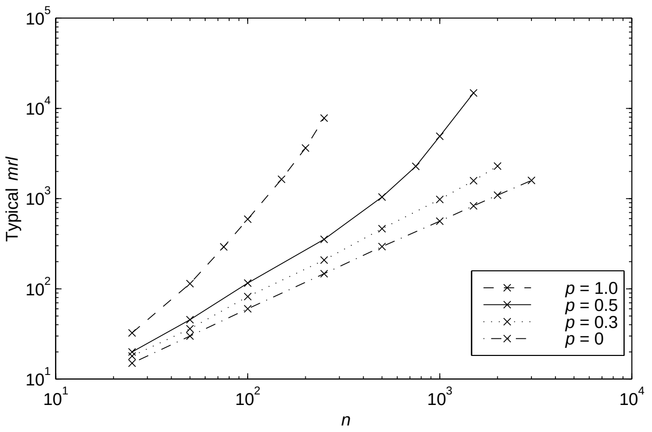

Figure 2. Typical cost, log-log scale

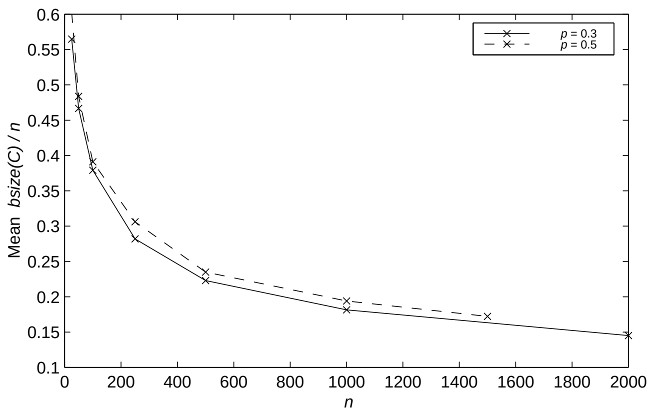

Figure 3. Average backbone size relative to n

cost to solve in the typical case. In this section we focus on the structural properties which might distinguish the satisfiable phase of $p = 0.3$ from that of $p = 0.5$.

One structural property of SAT instances which has been studied in Random SAT is the *backbone* [15, 17] . If the SAT instance C is satisfiable, the backbone is the set of literals which are logically entailed by C[5]. A literal x is entailed by C iff every solution to C makes x true. The backbone size $bsize(C)$ is the number of literals entailed by C. Parkes showed that backbone size is an important factor in cost for WSAT/SKC on Random 3-SAT at the threshold [17]. There was a strong positive relationship between backbone size and average cost. Cost increases with backbone size presumably because backbone size limits the regions of the search space in which solutions exist. A large backbone implies that solutions are tightly clustered whereas a small one allows them to be widely distributed.

Monasson *et al.* [15] claim that the backbone changes discontinuously as m/n is varied in the limit for $p > p_0$. Presumably for $p < p_0$ the change is continuous, reflecting the continuous change in the satisfiability probability. The possibly different nature of the backbone either side of p_0 could be one reason for the differences in local search cost. We calculated backbone sizes for the $p = 0.3, 0.5$ SAT instance collections[6]. The average $bsize(C)$ over each collection is plotted against n in Figure 3. The data shows that backbone sizes in the $p = 0.5$ satisfiable phase are slightly larger for this range of n, although we found no clear evidence of a qualitative difference between the distributions of $bsize(C)$ at these two values of p (as for example can be seen between $p = 0$ and $p = 1$ [15]).

As mentioned, $bsize(C)$ is an important factor partly determining the local search cost of an instance within a collection. However, there is also evidence that the difference in mean $bsize(C)$ between $p = 0.3$ and $p = 0.5$ is not the cause of the stark difference in typical cost. To establish this we focused on the "cost-typical" portion of each collection. We ranked each instance in the collection by local search cost as determined in Section 4. The cost-typical portion is the 10 % of the instance collection whose costs lie in the centre of this ranking. Table 3 gives statistics on the backbone sizes of the cost-typical portion instances at various settings of n and p.

n	$p = 0.3$		$p = 0.5$	
	Median $bsize$	Cost-typical *mrl* range	Median $bsize$	Cost-typical *mrl* range
25	15	17 – 19	16	19 – 21
50	24	35 – 38	25	43.5 – 48
100	38	79 – 85.5	39	109 – 126
250	72	198 – 217.5	73	327 – 381
500	108	437 – 490.5	95	936 – 1187
1000	176	1110.5 – 1339.5	148.5	3868.5 – 6156
1500	223.5	1462 – 1716.5	223.5	11621.5 – 18932.5

Table 3. Cost-typical instances for $p = 0.3, 0.5$. The collection sizes were made equal at each level of n. Note that the cost ranges do not overlap except for $n = 25$ and that the costs of cost-typical instances are quite narrowly distributed around the median.

The data suggest that for larger n, where the cost differences are greater, the backbones of the $p = 0.3$ cost-typical instances are generally no smaller than those at $p = 0.5$ even though they are much less costly. This would imply that backbone size is not the reason that typical cost grows so much more quickly for $p = 0.5$.

In a recent article [20], we showed that a large backbone combined with *backbone fragility* was a very plausible cause of the high typical local search cost which appears near $\alpha_{0.5}$ in Random 3-SAT. An instance is backbone-fragile if, when a few of its clauses are removed at random, much of the backbone is lost (i.e. many fewer literals are entailed). We suggested that backbone-fragile instances are difficult for local search because assignments at which few clauses are unsatisfied are so much more widely-distributed through the search space than solutions. If the backbone size changes discontinuously (as m/n is varied) we would expect to observe backbone fragility: the removal of a small number of clauses will cause a finite fraction of entailed literals to be lost. We studied the effect on the backbone of removing clauses from instances from the collections $p = 0.5, n = 1000$ and $p = 0.3, n = 1000$. We progressively removed clauses from each instance, calculating the change in backbone size as they were removed. Although the clauses were removed at random, this was

[5] Monasson *et al.* also extend their definition of the backbone to unsatisfiable instances, which we do not consider.

[6] The basic method for determining whether a literal x is entailed by C is to determine the satisfiability of $C \wedge \neg x$. This method for finding the backbone can then be extended by certain optimisations. If a solution to C is found making $\neg x$ true then x cannot be entailed. So for example, many entailed literals can often be eliminated by finding several solutions using local search. If x is entailed by C, C is equivalent to $C \wedge x$. This means that in future searches on C, branches where x is false can be pruned.

done so that the ratio of 2- to 3-clauses was preserved in each case.

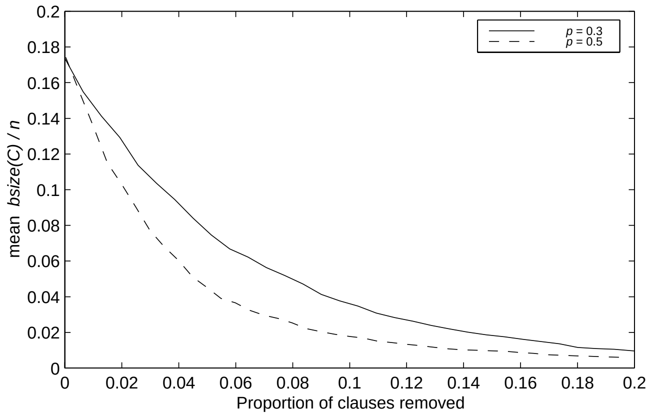

Figure 4. The effect on mean backbone size of removing clauses from cost-typical satisfiable threshold instances $n = 1000$.

Figure 4 shows the effect on backbone size of removing clauses from the cost-typical instances. The original backbone size (seen at 0 clauses removed) is very similar, but as we remove clauses the backbone decays much more sharply at $p = 0.5$ than at $p = 0.3$. The $p = 0.5$ instances are more backbone-fragile. This preliminary result combined with our previous work [20] suggests that backbone fragility may also be the cause of the change in cost scaling between $p = 0.3$ and $p = 0.5$. While we do not show data here, the same effect was also seen with respect to the number (rather than the proportion) of clauses removed. A similar effect is also seen when averaged over all satisfiable instances rather than just the cost-typical ones.

6 CONCLUSIONS AND FURTHER WORK

We tested the local search algorithm NOVELTY$^+$ at optimal noise levels on the Random 2+p-SAT distribution. The experimental data suggested some significant conclusions. Typical NOVELTY$^+$ cost appears to scale as a low polynomial up to $p = 0.3$. However, solving all instances using this algorithm appears to be costly for $p > 0$. For $p = 0.5$ typical cost appears to grow superpolynomially but more slowly than a simple exponential: in this sense cost scaling for $p = 0.5$ is similar to that for $p = 1$. The apparent cross-algorithm nature of the change in typical cost scaling from low polynomial to superpolynomial suggests that some 'hardness property' of instances becomes common enough to effect the median near p_0. We also reported on some experiments designed to identify the hardness property. The average backbone size is larger for $p = 0.5$ than it is for $p = 0.3$. However, for cost-typical instances and sufficiently large n, the backbone size is no smaller for $p = 0.3$ so this can be ruled out as the cause of high typical cost. Another possibility is that the cost-typical instances are more backbone-fragile for $p = 0.5$ than for $p = 0.3$ and this causes the high typical cost. Our initial experiments are consistent with this hypothesis.

Many interesting questions remain to be answered. How do other classes of SAT algorithm perform on Random 2+p-SAT? Do all algorithms begin to encounter difficulties at the same value of p? The hardness property should be identified in more detail. Does backbone fragility affect complete algorithms? If not, what is the property causing superpolynomial typical cost for complete algorithms? Can these properties be found in non-random, realistic SAT instances?

ACKNOWLEDGEMENTS

The first and second authors are members of the APES research group (`http://www.cs.strath.ac.uk/˜apes`). Josh Singer is supported by EPSRC studentship 97305799. We would like to thank Toby Walsh and other members of APES for helpful discussions and Allen van Gelder for making MODOC available.

REFERENCES

[1] D. Achlioptas, L.M. Kirousis, E. Kranakis, and D. Krizanc, 'Rigorous Results for (2+p)-SAT', in *Proceedings of RALCOM 97*, pp. 1–10, (1997).

[2] B. Aspvall, M. F. Plass, and R. E. Tarjan, 'A linear-time algorithm for testing the truth of certain quantified Boolean formulas', *Information Processing Letters*, **8**(3), 121–123, (1979).

[3] P. Cheeseman, B. Kanefsky, and W. Taylor, 'Where the Really Hard Problems Are', in *Proceedings of IJCAI-91*, pp. 331–340. Morgan Kaufmann, (1991).

[4] D. Clark, J. Frank, I. P. Gent, E. MacIntyre, N. Tomov, and T. Walsh, 'Local Search and the Number of Solutions', in *Proceedings of the Second International Conference on the Principles and Practice of Constraint Programming*, pp. 119–133. Springer, (1996).

[5] S. Cook, 'The Complexity of Theorem-Proving Procedures', in *Proc. 3rd Ann. ACM Symp. on Theory of Computing*, pp. 151–158, (1971).

[6] S. Cook and D. Mitchell, 'Finding Hard Instances of the Satisfiability Problem: A Survey', in *Satisfiability Problem: Theory and Applications*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, (1997).

[7] J. M. Crawford and L. D. Auton, 'Experimental Results on the Crossover Point in Random 3SAT', *Artificial Intelligence*, **81**, 31–57, (1996).

[8] A. Frieze and S. Suen, 'Analysis of Two Simple Heuristics on a Random Instance of k-SAT', *Journal of Algorithms*, **20**(2), 312–355, (1996).

[9] I. P. Gent, E. MacIntyre, P. Prosser, and T. Walsh, 'The Scaling of Search Cost', in *Proceedings of AAAI-97*, pp. 315–320. AAAI Press / The MIT Press, (1997).

[10] J. N. Hooker, 'Needed: An empirical science of algorithms', *Operations Research*, **42**(2), 201–212, (1994).

[11] Holger Hoos, *Stochastic Local Search - Methods, Models, Applications*, Ph.D. dissertation, Darmstadt University of Technology, 1998.

[12] Holger Hoos, 'On the Run-time Behaviour of Stochastic Local Search Algorithms for SAT', in *Proceedings of AAAI-99*, pp. 661–666. AAAI Press / The MIT Press, (1999).

[13] T. Larrabee and Y. Tsuji, 'Evidence for a Satisfiability Threshold for Random 3CNF Formulas', Technical Report UCSC-CRL-92-42, Jack Baskin School of Engineering, University of California, Santa Cruz, (1992).

[14] D. McAllester, B. Selman, and H. Kautz, 'Evidence for Invariants in Local Search', in *Proceedings of AAAI-97*, pp. 321–326. AAAI Press / The MIT Press, (1997).

[15] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, '2+P-SAT: Relation of Typical-Case Complexity to the Nature of the Phase Transition', *Random Structures and Algorithms*, **15**, 414 – 440, (1999).

[16] C. H. Papadimitriou, 'On selecting a satisfying truth assignment', in *Proc. 32nd IEEE Symp. on the Foundations of Comp. Sci.*, pp. 163–169, (1991).

[17] A. Parkes, 'Clustering at the Phase Transition', in *Proceedings of AAAI-97*, pp. 340–345. AAAI Press / The MIT Press, (1997).

[18] A. Parkes and J. Walser, 'Tuning Local Search for Satisfiability Testing', in *Proceedings of AAAI-96*, pp. 356–362. AAAI Press / The MIT Press, (1996).

[19] B. Selman, H. Kautz, and B. Cohen, 'Noise Strategies for Improving Local Search', in *Proceedings of AAAI-94*, pp. 337–343. AAAI Press / The MIT Press, (1994).

[20] J. Singer, I. P. Gent, and A. Smaill, 'Backbone Fragility and the Local Search Cost Peak', to appear, *Journal of AI Research*, (2000).

[21] A. van Gelder and F. Okushi, 'A propositional theorem prover to solve planning and other problems', *Annals of Mathematics and Artificial Intelligence*, **26**, 87–112, (1999).