Phase Transitions from Real Computational Problems *

Ian P. Gent
Department of Computer Science
University of Strathclyde
Glasgow, UK
ipg@cs.strath.ac.uk

Toby Walsh
Mechanized Reasoning Group
IRST, Trento &
DIST, University of Genoa, Italy
toby@irst.it

Abstract

We examine phase transitions in problems derived from real computational problems using a wide variety of algorithms. These phase transitions resemble those observed with randomly generated problems. Real problems do, however, contain new features (*e.g.* large scale structures rare in random problems) which can make them significantly harder than random problems. Our results suggest a new methodology for benchmarking algorithms. In addition, they help to identify the location of the really hard *real* problems.

1 Introduction

A conventional method for comparing the performance of algorithms is to use benchmark problems. Since the supply of benchmark problems is usually limited, we may be unable to perform a statistically significant comparison, or to determine accurately how performance depends on problem size and problem difficulty. An alternative approach is to use random problems which are cheap to generate at different problem sizes. Unfortunately random problems are typically easy to solve. For example, random k-colourable graphs are almost all easy to colour [22]. However, hard random problems can be found at a phase transition [2]. Such problems are frequently used to benchmark algorithms and to determine how performance scales with problem size [6; 4; 19]. One disadvantage of random problems is that they may not be representative of the kind of problems we are likely to meet in practice. For example, random problems may lack features that can make real problems very hard. In this paper, we show that it is possible to merge the two approaches. By means of four case studies, we identify phase transitions that occur in problems derived from real computational problems, and examine the change in problem difficulty as we traverse the phase boundary.

*The second author was supported by a HCM Postdoctoral fellowship, while the first author was in part supported by his wife, Judith Underwood. We thank Alan Bundy, the members of the Mathematical Reasoning Group at Edinburgh (supported by SERC grant GR/H 23610), Fausto Giunchiglia and the MRG group at IRST for their constructive comments and yet more CPU cycles. For software, benchmark problems and programming assistance, we thank Robert Craig, Ian Frank, Henry Kautz, Patrick Prosser, Gordon Reid, Peter Ross, Bart Selman, and Mark Stickel.

2 Traveling Salesman Problem

Given n cities, a tour length l and a matrix that defines the distance between each pair of cities, the traveling salesman decision problem (the TSP decision problem) is to determine if a tour of length l or less exists which visits all n cities. The TSP decision problem is one of the most famous NP-complete problems. To solve it, we use an implementation of the branch and bound algorithm written by Robert Craig at AT&T Bell Labs which uses the Hungarian algorithm for branching, minimally adapted by us to solve the decision rather than minimization problem. Branch and bound is one of the best complete algorithms for the TSP decision problem. We use standard benchmark data from TSPLIB [20], the capitals of the 48 contiguous states of the U.S.A.

In [2], Cheeseman *et al.* show that many NP-complete problems have a natural "order" parameter, that there is a phase transition as this parameter is varied[1], and that hard instances are often associated with this transition. For random TSP problems with n cities uniformly distributed over a rectangular area A, the expected optimal tour length [1] approaches the

[1]It has become the custom to discuss phase transitions in computational problems in terms of "order parameters". Such parameters can usually be more correctly called "control parameters" since they are externally varied by the user.

limit,

$$\lim_{n \leadsto \infty} \frac{l_{opt}}{\sqrt{n}} = k.\sqrt{A}$$

where $k \approx 0.75$. This suggests that a natural parameter for the TSP decision problem is $l/\sqrt{n.A}$. In [11], we confirm this suggestion by showing that for *randomly generated* TSP problems a phase transition occurs at a fixed value of $l/\sqrt{n.A}$.

In Figures 1 and 2, we plot two views of the phase transition as we vary this parameter using real data from TSPLIB. In Figure 1, we determine how computational expense changes as we search for tours of different lengths. We fix the number of capitals n at 24 and vary the tour length l from 500km to 15,000km in steps of 500km. We used the same 1000 sets of 24 capitals at each point. We plot results by contours of percentiles. For example, the 90% contour gives the cost that was exceeded by only 10% of other tests: in this case this would be the $100th$ most difficult problem. Figure 1 gives contours for best case, 50% (median), 90%, 99%, and 100% (observed worst case) for the number of nodes searched by the branch and bound algorithm. Note that we have plotted the *log* of the number of nodes searched so search cost varies over many orders of magnitude. All logs in this paper are to base 10. The dotted line represents the probability that a tour of the appropriate length or less exists. The phase transition, defined as the point at which 50% of problems have a tour, occurs at $l/\sqrt{n.A} \approx 0.56$. The worst case needed 2,933,071,577 nodes for a tour of length 8500km or less, in a region where 99.6% of the problems have a tour. This is over *five* orders of magnitude worse than the worst median of 5,096 nodes at tour length 7500km, where 31.1% of tours were possible and worst case behaviour was 232,077,515 nodes. It might be expected that these extraordinarily hard problems are insoluble, the result of exhaustive failed search. This is not so: the hardest insoluble problem in the whole run required only 35,401,612 nodes, an amount of search exceeded by no less than 16 soluble problems in the run. The occurrence of exceptionally hard problems has been reported in [13; 7] and is discussed later in this section.

In Figure 2, we determine how computational expense changes as we try to visit more capitals. We fix the tour length l at 7971km, which is $\frac{3}{4}$ the optimal tour length for all 48 capitals, and vary the number of capitals visited from 10 to 48 in steps of 2. At each point, we used 200 sets of capitals. The phase transition occurs at $l/\sqrt{n.A} \approx 0.56$. This is almost identical to the position of the phase transition in the previous experiment. The worst case is 163,150,184 nodes for a 26 capital tour, in a region where 52% of the problems have a tour. This is four orders of magnitude worse than the worst median problem.

As in the well-known phase transition for random satisfiability problems [17], there is a rapid phase transition in both experiments. In the soluble region, problems are under-constrained and thus typically easy. In the insoluble region, problems are over-constrained and again typically easy. In the phase transition inbetween, problems are critically constrained and typically hard. Although median problem difficulty peaks in the middle of phase transition, the hardest problems can occur in under-constrained regions where nearly 100% of problems are soluble. Extremely hard and under-constrained problems in soluble regions have also been observed for random graph-colouring and satisfiability problems [13; 7].

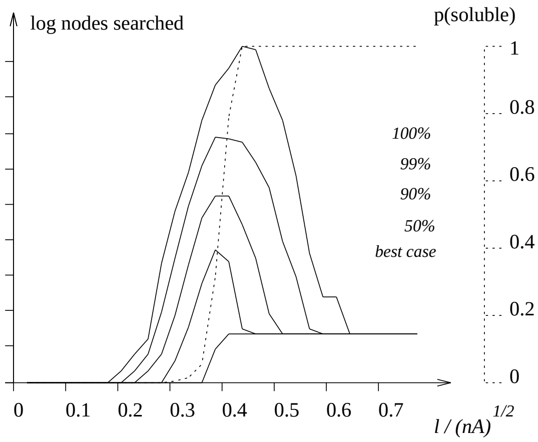

Fig 1: Varying tour length

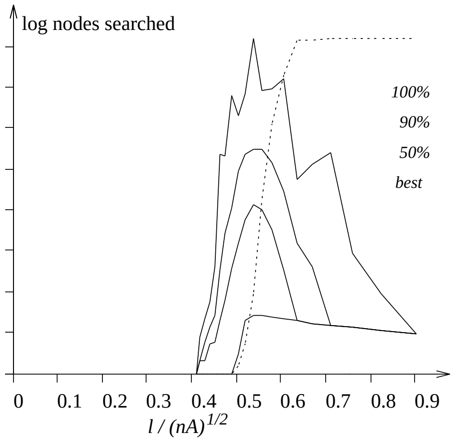

Fig 2: Varying number of capitals

As a comparison, we ran a set of experiments with randomly generated TSP decision problems in which cities were placed on a unit square of side 1000 at random. In Figure 3, we fix the number of randomly generated cities at 24 and vary the tour length from 250 to 7500 in steps of 250. We used the same 1000 sets of 24 random cities at each point. The phase transition occurs at $l/\sqrt{n.A} \approx 0.85$. The worst case in Figure 3 is 8,699,820 nodes in a region where 28.7% of problems have a tour. This is four orders of magnitude worse than the worst median problem, but still 300 times easier than the worst problem in the real data. Hard random problems also occur in the mostly soluble region. The worst such problem took 4,412,760 nodes in a region where 97.7% of problems (including this one) have a tour.

The phase transition for random data is very similar to that seen with real data. There are two significant differences. First, the real phase transitions occur at a smaller value of $l/\sqrt{n.A}$. Second, real problems are significantly harder than random problems. This may be a result of the different distribution of cities in randomly generated problems compared to the real data. If we normalize mean inter-city distance to 1 in each problem, the mean standard deviation in the distance matrix for the sets of 24 U.S. capitals was 0.586, while the mean standard deviation for the sets of 24 random cities was 0.469, considerably less. Cheeseman *et al.* [2] propose that the standard deviation in the cost matrix is a parameter for the TSP minimization problem (i.e. the problem of finding an optimal tour).

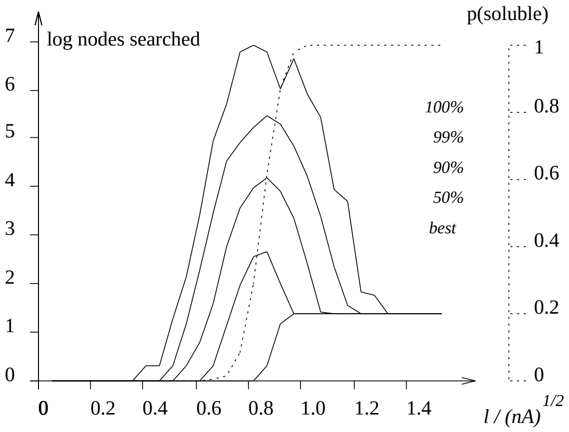

Fig 3: Random TSP problems

The phase transitions observed above result from a mixture of behaviours on soluble and insoluble problems. To isolate the different behaviour, we found the optimal tour for the first 100 sets of 24 U.S. capitals from our first experiment. We could then set the tour length to the decision problem to be a known distance, d from the optimal tour for each problem. At $d = 0$km only optimal tours will be found whilst at $d = -1$km no tours will be

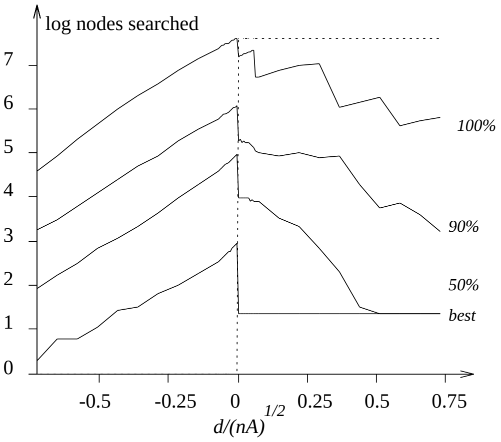

Fig 4: $l_{opt} + d$ for the U.S. capitals

found at all. By design, the phase transition thus occurs abruptly between $d = -1$km and 0km. This gives a clear picture of how problem difficulty is related to the distance from the individual transition in solubility. In Figure 4, we vary d, the distance from the optimal tour from -1000km (insoluble) to 1000km (soluble) in steps of 100km, and for finer detail from -100km to $+100$km in steps of 10km. In addition, we plot the cost at -1km. For each distance d, we tested 100 sets of capitals, with $l = l_{opt} + d$. This graph demonstrates two different types of behaviour. In the insoluble region, there is exponential growth as the phase boundary is approached. The log plot suggests search grows approximately as $2^{137.d/\sqrt{n.A}}$ in all contours: that is, an increase of 100km in the tour length roughly doubles search cost. This is surprisingly consistent behaviour considering the small problem size and the fact that this is real and not random data. In the soluble region, problems typically become harder as we approach the phase boundary since we can accept increasingly fewer sub-optimal tours. However, some of the hardest soluble problems have regions where search increases exponentially as we move *away* from the phase boundary. With these problems, poor branching decisions early on result in an exhaustive (and unsuccessful search) for a tour of given length.

As an example, the most difficult problem to optimize takes 14,875,150 nodes if a tour of its optimal length 7,995km is demanded. At the very start of search, the second leg attempted (and the sixth branching decision made) is from Raleigh NC to Columbus OH. Only much later in the search is it discovered that it would be much better to visit Hartford CT, Montpelier VT, and then Albany NY between Raleigh and Columbus. Because of such poor initial decisions, increasing the initial bound can make search *deeper* and thus more costly. In this example, demanding a tour 80km longer than the optimal increases search to 21,932,461 nodes since more search is needed to prove incorrect the Raleigh-Columbus leg. This increase is despite the fact that a tour 24km longer than the optimal is actually found in which Albany and Montpelier are visited in the reverse order, and Lansing MI is inserted between Montpelier and Columbus. Sudden drops in difficulty in the worst case contour follow the appearance of new and significantly longer sub-optimal tours.

To conclude, there is a clear phase transition in the TSP decision problem using real data. Whilst median difficulty displays a simple easy-hard-easy pattern, the hardest problems can occur in soluble regions following poor branching decisions early in search. The phase transition for random data looks very similar except that real problems are significantly harder than random problems. Our results clearly refute the claim of Kirkpatrick and Selman [15] that "there are other NP-complete problems (for example, the traveling salesman problem or max-clique) that lack a clear phase boundary at which 'hard problems' cluster". It would be interesting to see if optimization heuristics display similar phase transition behaviour.

3 Time-tabling

Our second problem class is exam time-tabling. Given n exams, k slots and s students, the exam time-tabling problem is to determine if there is an assignment of exams to slots such that no student takes two exams simultaneously. This is equivalent to deciding if there is a k-colouring of a graph with each node representing an exam, and each edge representing the constraint that a student is taking a particular pair of exams. The exam time-tabling problem is thus NP-complete. We consider two extensions of the basic problem. First, lecturers can exclude certain exams from certain slots. Second, there are four slots per day and students are not allowed to take exams in consecutive slots; these additional "friendliness" constraints typically make time-tabling harder.

We use the exam time-tabling problem for the AI Department at Edinburgh University, for the years 1992 to 1994. Exam time-tabling is naturally a binary constraint satisfaction problem (binary CSP) with variables representing exams, each of which needs to be assigned a slot. We therefore tried a recent CSP algorithm, Patrick Prosser's forward checking, conflict-directed back-jumping algorithm with directed k-consistency and a fail first heuristic for picking the next variable assignment [18]. Unfortunately, it has difficulty with our real exam time-tabling problems. On the complete 1994 problem with 59 exams, 36 slots and 200 students, Prosser's algorithm explored 4,213,045 nodes and performed 411,770,462 consistency checks.

Given recent success encoding other NP-hard problems into propositional satisfiability (SAT), we encoded the exam time-tabling problem into SAT and tried Mark Stickel's efficient implementation of the Davis Putnam procedure which uses a form of intelligent backtracking based on dependency checking to close open branches. This procedure was more successful and we were able to perform a complete phase transition experiment using the 1994 data. Although we could solve the full 1992 and 1993 problems without difficulty, we were unable to perform complete phase transition experiments for these years, despite the fact that there were fewer exams and students than in 1994. In the hard parts of these phase transitions, some problems take over a billion branches.

For a fixed number of exams, two natural parameters are the number of students and of slots. In Figure 5, we determine how computational expense changes as we add students using the 1994 data[2]. We fix the number of slots at 36 and vary the number of students from 10 to 200 in steps of 10. At each point except the last, we used 100 sets of students. Unlike all other figures in this paper, we plot the mean number of branches searched, and not on a log basis. The worst case is 4,227,072 branches for two unsatisfiable problems of time-tabling 170 and 150 students respectively. We will explain later the exact duplication in the number of branches. Corne *et al.* used a genetic algorithm[3] to construct the actual time-table for the 1994 exams. As they were unable to find a perfect time-table, one student had to sit two consecutive exams. Our results show for the first time that this was inevitable – there is no time-table for the full 200 students which meets all the constraints.

The phase transition when we vary the number of students is somewhat different to previous phase transitions. In particular, the hardest problems appear not in the middle of the phase transition but in the unsatisfiable region. These problems have very large proofs. To explore this further, we adapt the technique developed for random satisfiability problems in [10] of finding minimal unsatisfiable subsets. In this case, we look for minimal insoluble subsets. A minimal insoluble subset is an insoluble subset of the exam pairs taken by students in which no strict subset is insoluble. We ex-

[2] In practice, as we add students, other aspects of the problem might change like the number of slots and exams. For simplicity, we ignore such complications here.

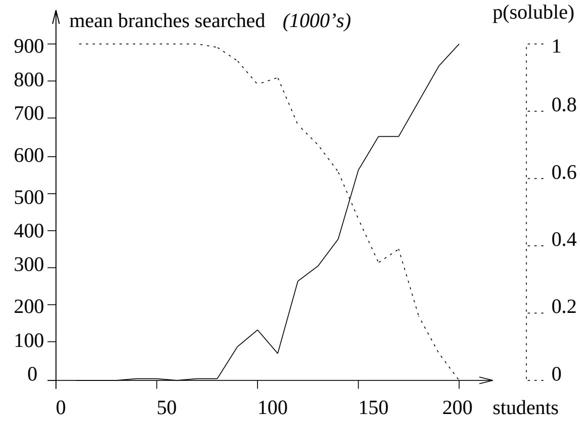

Fig 5: Varying students

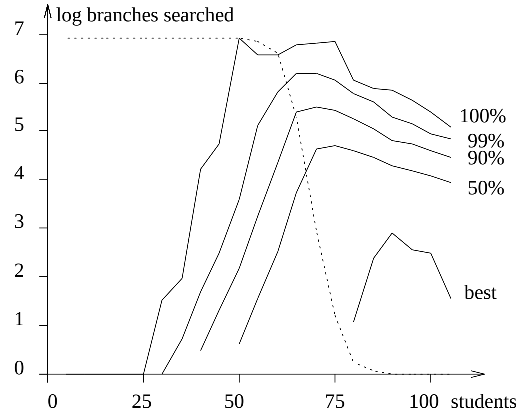

Fig 6: Random time-tables

tracted minimal insoluble subsets from the full 1994 problem. Each minimal insoluble subset we found was a 10-clique of exams which were prohibited from occurring in the afternoon. As there are only 9 mornings each with 2 slots, it is impossible to time-table the 10-clique given the friendliness constraints. The behaviour of the Davis Putnam procedure through the phase transition can be explained almost solely by the probability of including a 10-clique; this probability increases to 1 as the number of students increases to 200. There is surprisingly little variation in difficulty of proving insolubility once such a clique is included. This explains why we observed the same worst case at different numbers of students. In both our encoding into SAT and in the original CSP formulation of the time-tabling problem, showing that a 10-clique cannot be time-tabled into 9 mornings is time-consuming. To show the unsatisfiability of a typical minimal insoluble subset takes 337,920 branches with Stickel's Davis Putnam procedure, and 388,214 nodes and 3,064,718 consistency checks with Prosser's CSP algorithm. The additional constraints in the full problem appear not to help the branching heuristics used by both algorithms. More sophisticated procedures which take advantage of the natural symmetries and which give short pigeonhole proofs may perform much better.

We also determined how how computational expense changes as we try to time-table with fewer slots. We used the 1992 data which has 44 exams, 92 students and 28 slots. We varied the number of slots from which all exams are excluded from 0 to 28 in steps of 1. As this experiment proved very time-consuming, we dropped the friendliness constraints and only tested a single problem at each point. The phase transition in solubility appears to occur around 12-13 slots. Whilst problems with 13 or more slots were solved searching only one branch, the 11 and 12 slot problems were too difficult to solve. Smaller problems were insoluble; their search cost growing exponentially as the transition is approached.

As a comparison, we also ran a set of experiments with randomly generated time-tabling problems. In Figure 6, we generate problems with 18 exams and 30 slots and vary the number of students from 5 to 105 in steps of 5. Each student takes 4 exams chosen at random. Each exam is excluded from any given slot with probability 0.5. For comparison, in 1994, on average students took 5.9 exams and each exam was excluded from 13 of the 36 slots. The worst case is 8,639,312 branches in a region where 99.7% of the problems are satisfiable. This is two orders of magnitude worse than the worst median problem, which took 49,570 branches. The phase transition for random time-tables in Figure 6 is very similar to that seen with random problems from satisfiability and other NP-complete classes. However, it is quite different from the phase transition observed using the 1994 exam data. Note again that median problem difficulty peaks in the phase transition but the hardest problems occur in an under-constrained region where nearly 100% of problems are soluble.

To conclude, we observed new effects in the time-tabling phase transitions. These effects can be attributed to the presence of large scale structures like 10-cliques which are rare in randomly generated problems. Such large scale structures can make problems hard even if they are well away from the phase transition.

4 Boolean circuit synthesis

Our third NP-complete problem class is Boolean circuit synthesis. We are given a specification of the input/output behaviour of a Boolean function, and wish to construct a programmable logic array (PLA) which meets the specification, and which uses a given number of AND gates. A PLA is a two layer AND/OR circuit widely used in digital i/c design. This problem can be encoded into satisfiability [14] with variables representing the possible wiring decisions. Selman and Kautz use this encoding to benchmark various hill-climbing procedures for satisfiability [21]. They build circuits for adders, comparators and randomly generated Boolean functions. They report significantly better performance with hill-climbing procedures than with either the Davis Putnam procedure or integer programming. In turn, Kamath *et al.* found that integer programming is much better than the classic Quine McCluskey method for circuit synthesis based on prime implicants [14].

As in [21], we found that the Davis Putnam procedure was unable to synthesize complex circuits, whilst hill-climbing procedures like GSAT with random walk could. GSAT with random walk is a greedy hill-climbing procedure which with probability p flips the variable assignment which most increases the number of satisfied clauses, and with probability $1-p$ flips a variable in an unsatisfied clause at random. Search is restarted from a new random truth assignment every *Max-flips* flips. GSAT is a semi-decision procedure – it finds models but cannot show that a problem is unsatisfiable. We are therefore unable to perform a complete phase transition experiment. However, a "half-phase transition" experiment can be performed with semi-decision procedures. Such an experiment tests the procedure on satisfiable instances which become increasingly constrained as we approach the phase boundary.

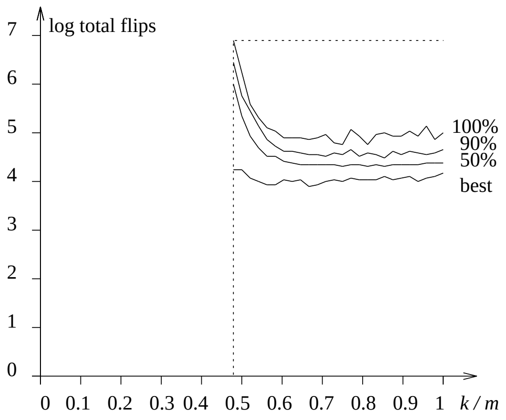

Fig 7: Circuit synthesis, 2-bit adder

If k is the number of AND gates, and m the number of 1's in the output specification, a natural parameter for Boolean circuit synthesis is the ratio k/m. If $k \geq m$ then circuit synthesis is rather trivial – we need merely read off from the truth table those inputs which compute 1, and connect each to a different AND gate. The

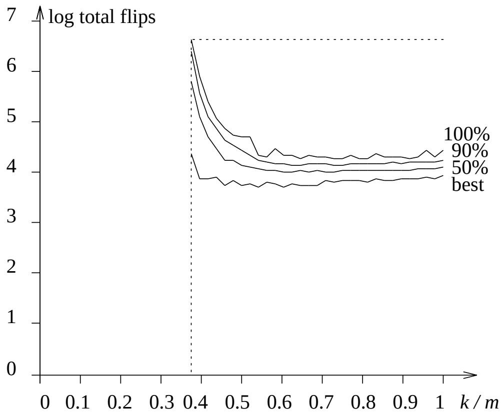

Fig 8: Circuit synthesis, random circuit

challenge is to build a circuit with fewer AND gates. In Figure 7, we plot one view of the half-phase transition as we vary k/m for a 2-bit full adder. This needs at least 23 AND gates to implement and has 48 1's in its output specification. We therefore varied the number of AND gates from 23 to 48 in steps of 1. At each point, we report the total number of flips performed by GSAT with random walk averaged over 100 runs. We always set *Max-flips* $=$ 1000N where N is the number of variables in the encoding into SAT and $p = 0.5$; these are usually good values for *Max-flips* and p [21; 12]. The worst case took 7,975,369 flips for the critically constrained problem of synthesizing the 2-bit adder from 23 AND gates. For randomly generated satisfiability problems, a phase transition occurs at a fixed ratio of clauses to variables [17; 9]. As a consequence of regularities in the encoding, this ratio is a poor predictor of the position of the phase transition for Boolean circuit synthesis. The ratio of clauses to variables at the phase transition is 17.0 for the 2-bit adder. The majority of these clauses contain 3 literals. For random satisfiability problems with 3 literals per clause, the phase transition occurs at just 4.3 clauses per variable [17].

As a comparison, in Figure 8 we plot a half-phase transition experiment for a randomly constructed Boolean function. This function has the same number of inputs and outputs as the 2-bit full adder, and a randomly generated truth table which has, like the adder, 48 out of the 96 output bits set to 1. The half phase transition for a random function is very similar to that seen with the adder. The main difference appears to be that, as random problems typically have more overlap in their truth tables, they require fewer AND gates to implement – the random circuit needed just 18 AND gates to implement. The ratio of clause to variables at the phase transition is now 15.0.

Previous studies of hill-climbing procedures like GSAT have used either hard random instances from the middle of the phase transition or benchmark problems. These graphs are thus the *first* illustration of the change in performance of GSAT as we approach the phase boundary for both real and random problems. The easiness of problems just a short distance from the phase boundary demonstrates the great care that needs to be taken when benchmarking algorithms. For example, the results of [14] must be treated with some caution as some of the problems used are far from the phase boundary. We speculate that much of the difficulty for GSAT of problems near the phase boundary can be attributed to the very small number of models compared to truth assignments. In studies with benchmark problems, we have observed that GSAT performs comparatively poorly whenever the number of models is low [12].

5 Boolean induction

Our fourth and final problem class is Boolean induction. We are now given only a *partial* specification of the input/output behaviour of a Boolean function, and must construct a programmable logic array which meets this partial specification, and which uses at most k AND gates. This corresponds to the "black box" problem of learning a Boolean circuit given some example inputs and outputs. We again encode the problem into satisfiability and use GSAT with random walk to find models. In all experiments, k is equal to the minimum number of AND gates needed to build the desired function. There is not therefore a phase transition in the probability of satisfiability since all problems are satisfiable. There is, however, a phase transition in the probability that the circuit constructed computes the correct function for *all* possible inputs. A simple parameter is σ, the ratio of input values in the partial specification to the total number of possible inputs. If $\sigma = 1$, then the probability that the circuit constructed computes the desired function is also 1. If $\sigma = 0$, then any circuit is a model, and the probability the the circuit computes the desired function will be small.

In Figure 9 we show the full phase transition as we vary σ for a 2-bit full adder using 23 AND gates. At each point, we report the total number of flips performed by GSAT with random walk averaged over 100 sets of randomly chosen inputs again using the standard parameter settings. The phase transition takes place very close to the limit of $\sigma = 1$. Median problem difficulty peaks as usual in the middle of the phase transition, but the hardest problems occur in a region where the probability of generating a correct circuit is *small*. Such problems are *under-constrained*. The worst case took 18,709,325 flips to synthesize the adder from 28 of 32 possible inputs. Interestingly, this was the only problem tested that generated a fully correct circuit from 28 or less inputs. We examined this problem in detail by testing GSAT on it for 100 restarts. It solved it only 30 times with *Max-flips* = 1000N = 2,231,000 flips. Of those 30 successes, 6 produced correct circuits. As a control, we repeated the experiment on the first random set of 28 inputs. This problem was much easier, the worst case being better than the best case of the previous problem. A circuit was generated successfully on each of the 100 restarts but *none* was the intended 2-bit adder.

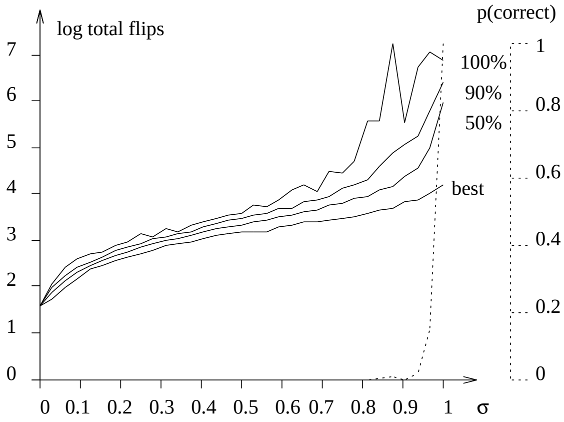

Fig 9: Boolean induction, 2-bit adder

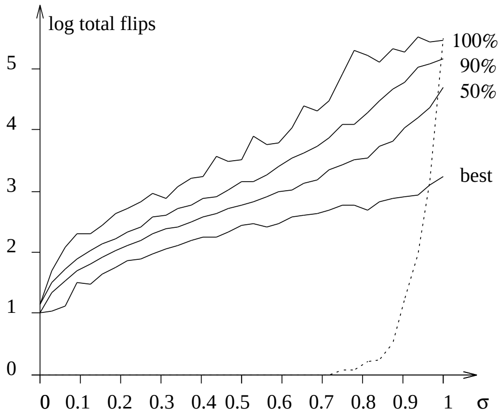

Fig 10: Boolean induction, 3-bit >

Similar behaviour is seen in Figure 10, giving results for the less complex 3-bit greater than comparator using 7 AND gates. The worst case was 321,661 flips on an under-constrained problem where 60 of the 64 inputs were specified: this was one of only 36% of problems at this point to generate a circuit with 100% accuracy. We also examined the phase transition for the induction of the random Boolean circuit of Figure 8, with results similar to Figure 9.

This example illustrates that phase transitions occur in macroscopic properties other than solubility. The phase transition here is in the quality of the solution, the probability that we construct the correct circuit. Indeed, there *cannot* be a phase transition in the probability of solubility since there is always a sufficient number of AND gates to ensure that problems are soluble. These graphs illustrate how incomplete procedures like GSAT can be used to perform full phase transition experiments. In our previous examples, over-constrained problems are insoluble. Here, over-constrained problems are soluble; the surplus of constraints merely guarantee that we generate the correct circuit. For the first time, we see GSAT performing more poorly on under-constrained problems than problems from the middle of the phase transition.

6 Related Work

Much recent interest in phase transition phenomena can be traced back to a seminal paper by Cheeseman *et al.* [2]. This paper identifies "order" parameters for graph colouring, satisfiability, Hamilton circuits, and the TSP minimization problem, and shows that for random problems there is a phase transition at a critical value of these parameters and that hard instances occur around the phase transition.

Tadd Hogg and Colin Williams [13], and independently the authors [7; 8; 9] explore in detail the distribution of problem difficulties through the phase transition for a variety of random problems including graph colouring, satisfiability, and the independent set problem. The results here support the conjecture made in [13] that "in practice, it is entirely possible that the kinds of troublesome problems we have identified [in the soluble region] might be encountered more frequently than the random graph ensemble would suggest". Whilst it is sometimes very difficult to find hard problems in the satisfiable region with random data [7], such problems appear to turn up frequently in real data. As another example, Claude Le Pape and Philippe Baptiste report benchmark job-shop scheduling problems in which sub-optimal solutions are more expensive to find than optimal solutions [personal communication, 1994], precisely the behaviour we analyze in detail in the TSP problem in §2.

James Crawford and Andrew Baker [5], express concern over the value of results on random satisfiability problems for real applications. They report results from encoding Sadeh's benchmark scheduling problems into satisfiability. They suggest that "scheduling problems . . . are much larger and much less constrained than the randomly generated theories generally studied." We note, however, that Sadeh's benchmark problems are *randomly* generated. Real scheduling problems, like the 1994 exam time-tabling problem, can be so tightly constrained that they have no complete solution.

Gary Lewandowski and Anne Condon [16] find a real time-tabling problem more useful than random data when comparing graph colouring algorithms. Frustrated by the difficulty of collecting real data, they build a random generator which attempts to model the features found in real data. The problem with this methodology is identifying the important features whilst maintaining the realism of the generated problems.

Finally, Corne *et al.* use a genetic algorithm to time-table exams for the Dept of AI at Edinburgh University [3]. It is difficult to access the performance of this algorithm since the complete examples they solve are quickly solved by encoding into satisfiability. By comparison, our phase transition experiments produced more challenging problem instances.

7 Conclusions

We have identified phase transitions in problems derived from *real* computational problems. These phase transitions are similar to those observed with randomly generated problems. For example, although median problem difficulty usually obeys a simple easy-hard-easy pattern with the most difficult median problem in the middle of the phase transition, the *hardest* problems can occur away from the phase transition in under-constrained regions. Both these phenomena have previously been observed with random problems [2; 7; 13]. These similarities demonstrate the value of previous empirical and theoretical research into random problem classes. Real problems do, however, introduce new phenomena. For instance, real problems can contain large scale structures rare in random problems. Real problems can therefore be significantly harder than similarly sized random problems.

Our experiments have considered a wide variety of computational problems. In the TSP experiments, we used real geographical data; this lacks much regular structure. In the time-tabling experiments, we used real exam time-tables; these have some regular structure due to the presence of modules and course pre-requisites. In the Boolean synthesis and induction experiments, we used real combinatorial circuits; these have a lot of regular structure reflecting the recursive nature of the functions being computed. We also used a wide variety of complete and incomplete algorithms. This suggests that our results are likely to be widely applicable.

Several of our experiments demonstrate phenomena that, as far as we aware, have not been previously reported even with random data. For example, we observe a phase transition in the quality of the solution, a macroscopic property different to the usual probability of solubility. In addition, we identify a problem class where GSAT performs more poorly on underconstrained problems than problems from the middle of the phase transition.

Many questions are prompted by this research. What other macroscopic properties display computational phase transitions? What other "hard" features occur in real problems, but rarely in our random problems? We encourage other researchers to address these questions, and to investigate computational phase transitions for further classes of real problems. Earlier research has answered the question, where are the really hard *random* problems? We can now start to answer the question, where are the really hard *real* problems?

References

[1] J. Beardwood, J.H. Halton, and J.M. Hammersley. The shortest path through many points. *Proc. of the Cambridge Philosophical Society*, 55:299–327, 1959.

[2] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proc. of the 12th IJCAI*, pages 163–169. 1991.

[3] D. Corne, H-L. Fang, and C. Mellish. Solving the modular exam scheduling problem with genetic algorithms. In *Industrial and Engineering Applications of AI and Expert Systems: Proc. of the 6th Int. Conference*, 1992.

[4] J.M. Crawford and L.D. Auton. Experimental results on the crossover point in satisfiability problems. In *Proc. of the 11th National Conference on AI*, pages 21–27. AAAI Press/MIT Press, 1993.

[5] J.M. Crawford and A.D. Baker. Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems. In *Proc. of 12th National Conference on AI*, pages 1092–1097. AAAI Press/MIT Press, 1994.

[6] I. P. Gent and T. Walsh. Towards an Understanding of Hill-climbing Procedures for SAT. In *Proc. of the 11th National Conference on AI*, pages 28–33. AAAI Press/MIT Press, 1993.

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

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

[9] I. P. Gent and T. Walsh. The SAT phase transition. In *Proc. of ECAI-94*, pages 105–109, 1994.

[10] I. P. Gent and T. Walsh. The satisfiability constraint gap. Research Paper 702, Dept. of Artificial Intelligence, Edinburgh, June 1994. To appear in *Artificial Intelligence*.

[11] I. P. Gent and T. Walsh. The TSP phase transition. Presented at *First International Joint Workshop on AI and OR*, Timberline, Oregon, June 1995.

[12] I. P. Gent and T. Walsh. Unsatisfied variables in local search. In editor J. Hallam, *Hybrid Problems, Hybrid Solutions*, pages 73–85. IOS Press, 1995. (Proc. of AISB-95).

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

[14] A.P. Kamath, N.K. Karmarkar, K.G. Ramakrishnan, and M.G.C. Resende. An interior point approach to Boolean vector function synthesis. In *Proc. of the 36th MSCAS*, 1993.

[15] S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, May 27 1994.

[16] G. Lewandowski and A. Condon. Experiments with parallel graph coloring heuristics. In *Proc. 2nd DIMACS Challenge*, 1993.

[17] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proc. of 10th National Conference on AI*. AAAI Press/MIT Press, July 1992.

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

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

[20] G. Reinelt. TSPLIB - a traveling salesman problem library. *ORSA Journal on Computing*, 3:376–384, 1991.

[21] B. Selman, H. Kautz, and B. Cohen. Noise Strategies for Improving Local Search. In *Proc. of 12th National Conference on AI*, pages 337–343, 1994.

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