The Constrainedness of Arc Consistency*

Ian P. Gent, Ewan MacIntyre, Patrick Prosser, Paul Shaw, and Toby Walsh

The APES Research Group, Department of Computer Science, University of Strathclyde, Glasgow G1 1XH, United Kingdom.
Email {ipg,em,pat,ps,tw}@cs.strath.ac.uk

Abstract. We show that the same methodology used to study phase transition behaviour in NP-complete problems works with a polynomial problem class: establishing arc consistency. A general measure of the constrainedness of an ensemble of problems, used to locate phase transitions in random NP-complete problems, predicts the location of a phase transition in establishing arc consistency. A complexity peak for the AC3 algorithm is associated with this transition. Finite size scaling models both the scaling of this transition and the computational cost. On problems at the phase transition, this model of computational cost agrees with the theoretical worst case. As with NP-complete problems, constrainedness – and proxies for it which are cheaper to compute – can be used as a heuristic for reducing the number of checks needed to establish arc consistency in AC3.

1 Introduction

Following [4] there has been considerable research into phase transition behaviour in NP-complete problems. Problems from the phase transition are now routinely used to benchmark algorithms for constraint satisfaction, satisfiability and other NP-complete problems. Phase transition behaviour has even suggested new heuristics for NP-complete problems [7]. Many interesting questions are raised by this research. Are phase transitions important in other complexity classes? If so, do they behave like phase transitions in NP? Does performance at the phase transition agree with the worst case complexity? Can we use phase transition behaviour to suggest heuristics for problems in these new complexity classes? In this paper we show that the same techniques used to study phase transitions in NP-complete problems can be used to study a phase transition in a polynomial problem class: establishing arc consistency in constraint satisfaction problems.

2 Constraint satisfaction

A binary constraint satisfaction problem consists of a set of variables V and a set of constraints C. Each variable $v \in V$ has a domain of values of size m_v.

* The authors are supported by EPSRC awards GR/L/24014 and GR/K/65706, and the EU award EU20603. The authors wish to thank other members of the APES research group for their help, and Gene Freuder.

Each binary constraint $c \in C$ rules out some proportion p_c of combinations of values for a pair of variables. We call p_c the "tightness" of a constraint. Two variables are adjacent if a constraint acts between them. The constraint satisfaction decision problem is then to determine if there exists an assignment of values to variables such that none of the constraints are violated. Consistency techniques are often applied to simplify such decision problems either before or during search. Arc consistency (or AC) is the simplest and most commonly used such technique. A problem is arc consistent if all values in all variables are *supported*. A value i for variable v is supported if, when i is assigned to v, all variables adjacent to v can be assigned values without violating constraints on v. Any value which is not supported cannot occur in a solution and can be removed. An arc consistency algorithm achieves an arc consistent state by repeatedly removing unsupported values. If it succeeds, we have established AC. If not, a domain wipe out occurs, where one variable has all values in its domain removed and the problem is insoluble. Therefore, the arc consistency decision problem is to determine if there exists a non-empty domain of supported values for each variable. The arc consistency algorithms studied here are AC3 [12] and AC6 [2]. The worst case complexity of AC3 is $O(em^3)$ [13] and of AC6 is $O(em^2)$, where m is the size of the largest domain and e is the number of edges in the constraint graph.

3 Phase transitions in NP

Phase transition behaviour has been studied in many NP-complete problems [4, 14, 9]. To unify such studies, [7] defines the constrainedness, κ of an ensemble of combinatorial problems as,

$$\kappa =_{\text{def}} 1 - \frac{\log_2(\langle Sol \rangle)}{N} \tag{1}$$

where N is the base 2 logarithm of the size of the state space, and $\langle Sol \rangle$ is the expected number of these states that are solutions. Since $0 \leq \langle Sol \rangle \leq 2^N$, κ lies in the range $[0, \infty)$. If $\kappa = 0$ then $\langle Sol \rangle = N$. Problems here are under-constrained since every state is expected to be a solution. If $\kappa = \infty$ then $\langle Sol \rangle = 0$. Problems here are over-constrained since no states are expected to be solutions. If $\kappa \approx 1$ both soluble and insoluble problems can occur. As problems are on the "knife-edge" between solubility and insolubility, it is often difficult to find solutions or prove that none exist.

This definition of constrainedness captures parameters used to study phase transitions in a wide variety of NP-complete problems including constraint satisfaction [6], satisfiability [14], graph colouring [4] and number partitioning [9]. As we vary problem size, the location of the phase transition tends to occur over a small range of κ. Other parameters can be less stable. For example, the expected number of solutions, which is used to predict the location of the phase transition in constraint satisfaction in [20], can grow exponentially with problem size. In random 3-SAT problems, $\langle Sol \rangle$ at the phase transition grows as $2^{0.18N}$ [8].

In this paper, it will be convenient to express κ as,

$$\kappa = -\frac{\log \rho}{N} \tag{2}$$

where $\rho = \langle Sol \rangle / 2^N$ is the *solution density*. That is, the probability that an arbitrarily chosen candidate in the ensemble is a solution.

4 Phase transitions in P

The same methodology developed to study phase transitions in NP-complete problems can be applied to polynomial problems. This yields several immediate results. First, the measure of constrainedness of a particular polynomial problem is the same as that for NP-complete problems. Consequently we are able to observe experimentally scaling of computational cost, and this is consistent with the theoretical worst-case complexity. Finally we use the definition of constrainedness to design new heuristics and explain the performance of existing heuristics.

There is an obvious complexity peak in graphs of the performance for the AC3 and AC6 algorithms in [2]. However, the phase transition in arc consistency was not systematically studied till [10]. For example, in Figure 1 of [10] we see a transition from a region where problems do not benefit from arc consistency, to one which can be proved insoluble by applying arc consistency. In between is a region of problems whose domains get smaller when arc consistency is applied, and which tend to be the hardest to make arc consistent. Graphs in [10] are plotted against κ_{csp}, the constrainedness of the constraint satisfaction decision problem.

Unlike NP-complete problems, the location of the phase transition in establishing arc consistency does not occur around some fixed value of κ_{csp} close to 1. For example, Table 1 of [10] reports the location of the phase transition shifting from $\kappa_{csp} \approx 1.08$ to $\kappa_{csp} \approx 3.68$. This might suggest a different approach is needed to locate phase transitions in polynomial problems compared to NP-complete problems.

In the rest of the paper, we show that the phase transition in establishing arc consistency is in fact very similar to that in NP-complete problems. The problem with the presentation of results in [10] is that κ_{csp} is the constrainedness of the NP-complete decision problem: Is there a consistent assignment of values to variables? But we are merely solving a polynomial problem, establishing arc consistency, and κ_{csp} does not measure the constrainedness of this problem. As soon as we compute the constrainedness of establishing arc consistency, we see very similar phase behaviour in P as in NP.

5 Constrainedness of arc consistency

To compute the constrainedness of establishing arc consistency, κ_{ac}, we need to decide what the state space, S, and a solution within it look like. A solution is a constraint satisfaction problem with arc consistent domains. Each point in the state space represents a constraint satisfaction problem with variables that have domains that are some subset of the original domains. The number of possible subsets of the domain of v is 2^{m_v}. The size of the state space is therefore $\prod_{v \in V} 2^{m_v}$, and hence $N = \sum_{v \in V} m_v$.

We next calculate the probability q that a candidate state is arc consistent. Assume that a constraint c between variables x and y is represented by a conflict matrix of size m'_x by m'_y. The candidate is arc consistent if there is at least one allowable value in each row and each column of every conflict matrix. To simplify the computation of the probability, q we assume independence between the probability that there is an allowable value in each row and the probability that there is an allowable value in each column. Whilst such an assumption is strictly false, similar independence assumptions have proved very successful in predicting the location of phase transitions in NP-complete problems. For example, in number partitioning, assuming independence between the binary digit positions predicts the location of the phase transition to within a 4% accuracy [9].

When we know the tightness of each constraint, it is easy to calculate if there is at least one allowable value. Our ensemble of problems has random independent constraints, of tightness p_c. Note that, on average, all arc consistency candidates for this problem class have constraints that are the same tightness as the corresponding constraints in the original problem. Thus, the values of p_c for each constraint c in the original problem can be used when assessing candidates. It follows that,

$$q = \prod_{c \in C} (1 - {p_c}^{m'_x})^{m'_y} (1 - {p_c}^{m'_y})^{m'_x}$$

To derive ρ, we need the mean value of q over the state space: $\rho = 2^{-N} \sum_{s \in S} q_s$. Due to variations in domain sizes over each candidate, this is not a simple calculation. Instead, we estimate ρ using a mean field approximation, and assume that all candidates have an equal value of q, derived from an "average" candidate with domain sizes half that of the original problem. This gives,

$$\rho = \prod_{c \in C} (1 - {p_c}^{m_x/2})^{m_y/2} (1 - {p_c}^{m_y/2})^{m_x/2}$$

That is,

$$\kappa_{ac} = \frac{- \sum_{c \in C} m_x \log_2(1 - {p_c}^{\frac{m_y}{2}}) + m_y \log_2(1 - {p_c}^{\frac{m_x}{2}})}{2 \sum_{v \in V} m_v} \tag{3}$$

In the remainder of this paper, unless otherwise indicated, we use regular problems with uniform domain size m, exactly $p_1 n(n-1)/2$ constraints between the variables, each of which has the same tightness p_2. These problems are categorised by the tuple of parameters $\langle n, m, p_1, p_2 \rangle$. We can then simplify κ_{ac} to:

$$\kappa_{ac} = -\frac{1}{2} p_1 (n-1) \log_2(1 - {p_2}^{m/2}) \tag{4}$$

6 Arc consistency phase transition

To test this new parameter we ran experiments on establishing arc consistency with $\langle n, m, p_1, p_2 \rangle$ problems with a domain size $m = 10$ and a varying number of variables n. We fix the average degree of problems at 5 by setting $p_1 = 5/(n-1)$ and at each value of n change p_2 in steps of 0.01. In Figure 1, we test 1000 randomly generated problems at each value of p_2, measuring the probability of establishing AC.

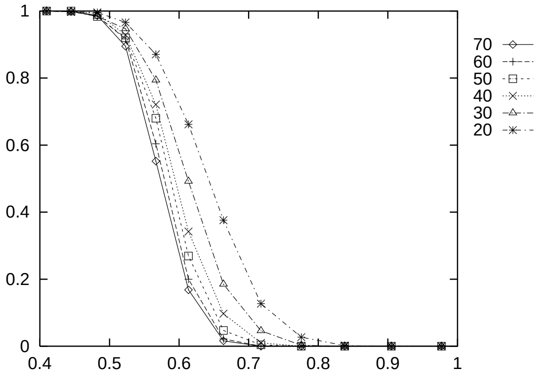

Fig. 1. Probability of establishing AC (y-axis) against κ_{ac} (x-axis) for varying n

As with NP-complete problems, a complexity peak for the cost of establishing arc consistency is associated with this probability phase transition. In Figure 2 we plot the computational cost (in terms of consistency checks) for AC3, establishing arc consistency for the same set of problems as in Figure 1. There is a familiar easy-hard-easy pattern. For $\kappa_{ac} \ll 1$, problems are under-constrained and it is easy to find an arc consistent state. For $\kappa_{ac} \gg 1$, problems are over-constrained and it is easy to observe domain wipe out. The hardest problems tend to occur in the phase transition in between when $\kappa_{ac} \approx 1$.

The complexity peak for AC6 can be observed at similar values of κ_{ac}. Bessière [2] reports experiments on problems $\langle 20, 5, 0.3, p_2 \rangle$, with p_2 varying in steps of 0.05, with 10 problems at each value of p_2. In Figure 4 (in [2]) the complexity peak for AC3 and AC6 occurs at $0.45 < p_2 < 0.5$ corresponding to $0.6 < \kappa_{ac} < 0.8$. And in Figure 5 (again in [2]) the complexity peak for $\langle 12, 16, 0.5, p_2 \rangle$ occurs at $p_2 = 0.8$ corresponding to $\kappa_{ac} = 0.73$.

Schiex *et. al.* established arc consistency with a non-uniform class of problems in which domain size varied randomly between 5 and 25, solving 20 problems at each value of constraint tightness [18]. They observed that "There is no clear "wipe-out" threshold as in the usual [fixed domain size] model" (page 221 of

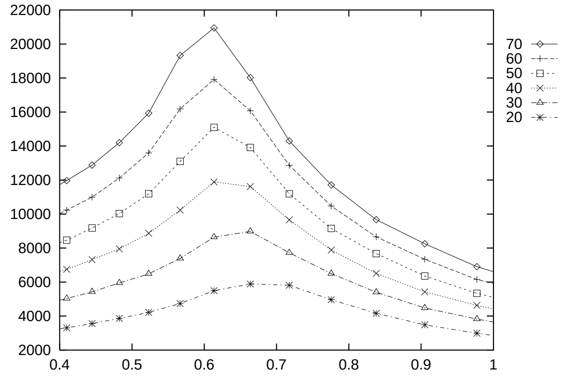

Fig. 2. Median consistency checks used by AC3 (y-axis) against κ_{ac} (x-axis) for varying n.

[18]). We performed experiments to determine if their observation holds true when we classify problems with respect to κ_{ac}.

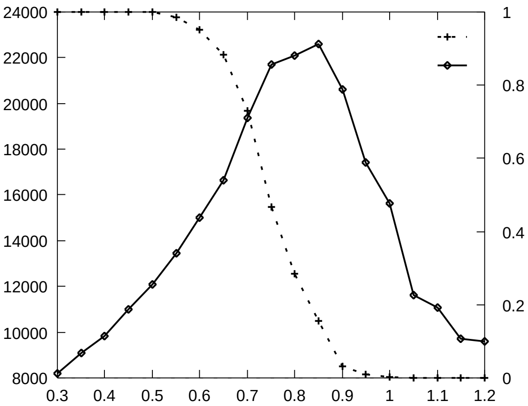

Fig. 3. Problems with non-uniform domain sizes. Y-axis on left and bold contour is computational cost, Y-axis on right and broken contour is probability of establishing AC. X-axis is κ_{ac}.

Problems were generated with $n = 20$ and $p_1 = 0.5$. Half of the variables were randomly chosen to have a domain size of 10, with the rest having a domain size of 20. Constraint tightness, p_2, was then varied from 0.01 to 0.99 in steps of 0.01, with 1000 problems at each point. When a problem was generated its κ_{ac} value was then computed and AC3 was applied. Figure 3 shows a clear phase transition in this non-uniform problem class, again at $\kappa_{ac} \approx 1$. The reason why this was not evident in [18] is due to variation in constrainedness of instances within the ensemble. The contribution to κ_{ac} (in Equation (3)) by a single constraint is sensitive to the tightness of the constraint and size of the domains involved in that constraint.

7 Finite size scaling

In NP-complete problem classes, the technique of finite size scaling has been borrowed from statistical mechanics [1] to model the change in the shape of the phase transition as problem size increases [11]. Around some critical value, κ_c, problems are indistinguishable except for a simple change in scale modelled by a power law. Finite-size scaling also works for this polynomial class. Following [7], we define a rescaled parameter

$$\gamma =_{\text{def}} \frac{\kappa - \kappa_c}{\kappa_c} N^{1/\nu} \tag{5}$$

$(\kappa - \kappa_c)/\kappa_c$ plays the same role as the reduced temperature, $(T - T_c)/T_c$ in a thermodynamic system whilst $N^{1/\nu}$ provides the scaling with problem size. As in [7] we define problem size as the number of bits needed to represent a state. In this case, we have $N = nm$. The values κ_c and ν are found by analysis of the empirical data using the methodology outlined in [6]. This gives $\kappa_c = 0.45$ and $\nu = 3.0$. Figure 4 shows the same data as Figure 1 rescaled by plotting against γ. This graph shows that finite size scaling successfully models the AC phase transition.

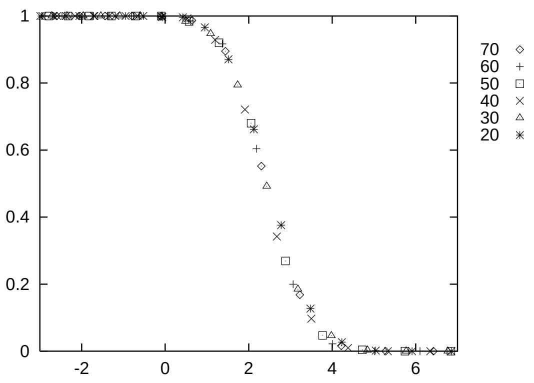

Fig. 4. Probability of establishing AC (y-axis) against γ for varying n with $\kappa_c = 0.45$, $\nu = 3.0$ (x-axis)

The rescaled parameter γ has been used to model growth of search cost as well as changes in probability as size increases in NP-complete problems [19, 6]. Rescaling of the number of checks performed by AC3 also gives a simple and accurate model of computational cost across the phase transition, even though AC3 is a polynomial algorithm. Furthermore this model gives close agreement with previous theoretical results. As the worst case complexity of AC3 is $O(em^3)$, and as the number of edges in the constraint graph, e is proportional to n in this study and m is fixed, computational cost should grow linearly with n. Accordingly we perform linear regression on the median checks performed by AC3 from $\gamma = -3$ to 7 in steps of 0.25, interpolating on observed data where necessary. Figure 5 shows that this linear model fits the data of Figure 2 very well. Note that the lines do *not* join points directly. Instead they join the values

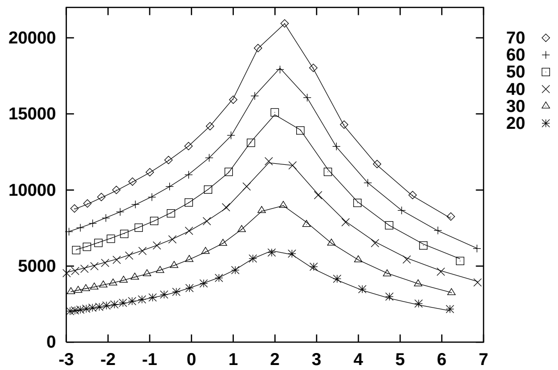

Fig. 5. Median consistency checks used by AC3 (y-axis) against γ (x-axis) for varying n. Points represent observed data while lines join modelled values of the data using linear regression.

modelled by linear regression. For example at $\gamma = 0$ the model is that checks $\approx 189n - 167$ while at $\gamma = 2$ the model is $297n + 87$. The closeness between the lines and points indicate how accurately linear scaling models computational cost. Note that the highest costs occur at $\gamma \approx 2$, very close to the point where 50% of problems could be made arc consistent and 50% could not. This correlation has been noted many times in NP-complete classes.

We also investigate scaling when we fix the number of variables $n = 20$ and vary the domain size. We use a constraint tightness $p_1 = 1$ so that the constraint graphs were cliques, and vary p_2 at each value of m. The probability of establishing AC shows a clear phase transition as κ_{ac} varies. We again rescale this data using finite size scaling with $N = nm$, $\kappa_c = 0.89$ and $\nu = 1.5$. Figure 6 shows that this models the phase transition very well.

We also modelled the growth in computational cost as the domain size varies.

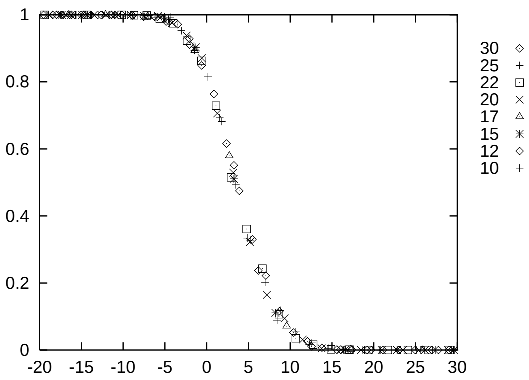

Fig. 6. Probability of establishing AC (y-axis) against γ for varying m with $\kappa_c = 0.89$, $\nu = 1.5$ (x-axis)

We no longer expect linear growth because the theoretical worst case is cubic in m. Since there are always 190 edges in our graphs, we ignored e in our models. We first attempted to model checks used as am^b. This gives a reasonable fit to the data and the peak values of b are close to 3, suggesting cubic growth. To give an explicit comparison between quadratic and cubic growth, we next investigated the model $am^2 + bm^3$ with least square linear fitting using singular value decomposition. This gives a very good fit to the data, as can be seen in Figure 7. Throughout the range of γ shown, our model suggests cubic growth. For example for underconstrained problems at $\gamma = -20$ the model is that checks $\approx 43.1m^2 + 2.26m^3$. For overconstrained problems at $\gamma = 20$ the model is $25.8m^2 + 8.37m^3$, while for critically constrained problems at $\gamma = 5$ the model is $60.0m^2 + 10.8m^3$. Grant and Smith [10] rule out the model em^3 but incorrectly conclude that growth is therefore nearer quadratic than cubic. Our model suggests that, with an appropriate scaling constant, growth is $O(em^3)$ which is the theoretical worst case. The coefficient of the cubic term suggests that problems at the phase boundary are significantly harder to solve than problems away from the boundary.

As the theoretical worst case behaviour seems to be attained, random problems appear to be able to contribute significantly to the study of the performance of polynomial algorithms like AC3. Interestingly, we observe scaling consistent with the theoretical worst case using just the median cost. This would suggest that the worst cases are not confined to rare and pathological examples. Problems at the phase transition may therefore be a valuable testbed for algorithms for AC and other polynomial problems, just as they are for NP-complete problems.

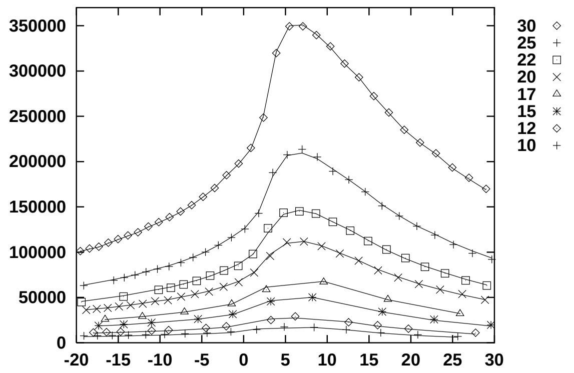

Fig. 7. Median consistency checks used by AC3 (y-axis) against γ (x-axis) for varying m. Points represent observed data while lines join modelled values.

8 Constraint Ordering Heuristics

At the heart of the AC3 algorithm is a set of directed constraints (often called arcs) waiting to be revised. The main loop of AC3 deletes an arc from this set and revises it: depending on the result other arcs may be added to the set if not already in it. This set of arcs is typically presented as a queue [12, 13, 21], such that arcs are removed in the order they were added, and propagation proceeds breadth first. For finite domains, we may use any other method for selecting the next arc to revise, and this opens up the scope for constraint ordering heuristics in AC3.

Wallace and Freuder performed a study on a number of heuristics based upon an intuitive ASAP (as soon as possible) principle where one attempts to prune domain values early [23]. They introduced heuristics based upon choosing (i) the arc with greatest constraint tightness, (ii) the arc for which the variable being checked for support has smallest domain size, (iii) an arc that will update a node which is involved in the most constraints. Heuristics (i) and (ii) worked well, reducing the number of checks by up to a factor of two over random selection or using a queue. Heuristic (iii) did not significantly reduced the number of checks over using a queue or random selection.

For NP-complete problems, the heuristic of making a choice that minimises the constrainedness of the resulting subproblem can reduce search over standard heuristics [7]. The intuition is that we want to branch on the most constrained variable into the least constrained and therefore most soluble subproblem. Similarly, for a polynomial problem like AC, we can use κ_{ac} as a constraint ordering heuristic in AC3. Here, the set of choices is the arcs in the current set maintained by AC3. We consider the remaining subproblem to have the same set of variables as the original problem, but with only those arcs still remaining in the set. We select the arc whose removal minimises the value of κ_{ac} of the remaining subproblem, ignoring the fact that subsequent revision of this arc may lead to new arcs being added to the set. By Equation (3), we choose the directed constraint c from variable x to variable y which has the maximal value of:

$$-m_x \log_2(1 - {p_c}^{m_y/2})$$

Even though this heuristic can significantly reduce the number of constraint checks need to establish arc consistency, it may not reduce runtimes. As with NP-complete problems, there are proxies for the heuristic of minimising constrainedness that are cheap to compute and that offer good performance. For instance, heuristics (i) and (ii) from [23] can be viewed as surrogates of the minimise-κ_{ac} heuristic, and we should expect them to perform well. Heuristic (i) chooses c such that p_c is maximised. Heuristic (ii) chooses c from x to y such that m_y is minimised. Everything else being equal, both these decisions reduce κ_{ac}.

Using the minimise-κ_{ac} heuristic, we performed experiments on the problem class $\langle 20, 10, 0.5\rangle$ with p_2 varied from 0 to 1 in steps of 0.01. Each point in figures 8 and 9 is the median value of 100 samples. For comparison with the minimise-κ_{ac} heuristic, we also implemented five variants of AC3 using a queue, a stack[2], picking a random element of the current set with no heuristic, and heuristics (i) and (ii) of [23]. We also report the number of revisions, *i.e.* the number

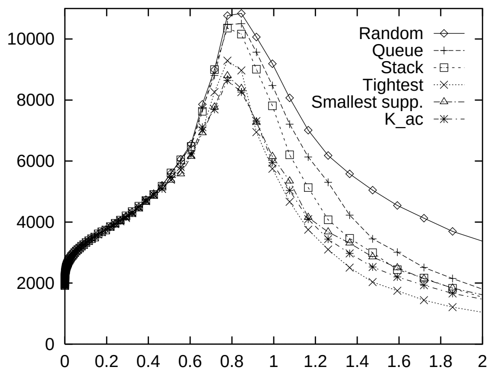

Fig. 8. Median number of checks performed by AC3 (y-axis) using various heuristics, against κ_{ac} (x-axis) for $\langle 20, 10, 0.5\rangle$ problems

of times an arc is taken out of the set. The number of revisions is important, since for problems involving structured (for instance arithmetic) constraints, or

[2] The results reported in previous sections used a stack and corresponds to a depth first propagation of constraints.

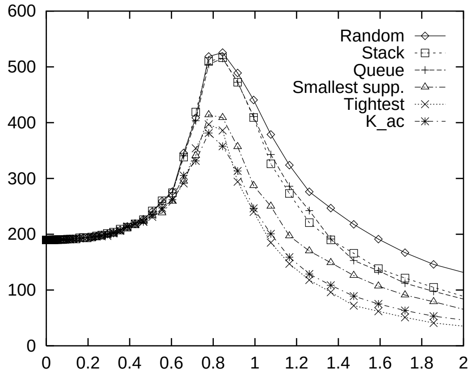

Fig. 9. Median number of revisions performed by AC3 using various heuristics (y-axis) against κ_{ac} (x-axis) for $\langle 20, 10, 0.5\rangle$ problems

when domains are represented as intervals, arc revisions may take constant time. In this situation, checks is a poor measure of computational effort.

At the phase transition, the κ_{ac} heuristic beats the other heuristics, whilst a random choice is always worst. What is important is that if we ignore the cost of the κ_{ac} heuristic we see that it performs well. If we then have to compromise the heuristic to bring down its cost we might very well have rediscovered Wallace and Freuder's heuristics. Heuristics like these deserve further attention, for example within search algorithms like maintaining arc consistency [17].

9 Related Work

There exist several polynomial problems in which phase transition behaviour has been seen. For example, random 2-SAT problems have a phase transition in satisfiability at a ratio of clauses to variables, L/N of 1 [5], and it has long been known that random graphs display sharp thresholds in properties like graph connectivity at critical values of the average degree [3].

Evidence for a phase transition in establishing arc consistency can be found in [2]. Bessière's graphs show clear peaks in the complexity of AC3 and AC6. Bessière's results suggest that AC4 [15], although of lower complexity, can perform poorly away from the complexity peaks of AC3 and AC6. An empirical study by Wallace reaffirms this; AC3 was nearly always better than AC4 [22].

The phase transition in arc and path consistency was first studied in depth in [10] with problems from the class $\langle 20, 10, p_1, p_2\rangle$. Using data from Table 1 in [10], we can calculate values for κ_{ac} for this phase transition. Unlike κ_{csp} at the phase transition which varied from 1.08 to 3.68, the phase transition occurs from $\kappa_{ac} \approx 0.6$ to $\kappa_{ac} \approx 1$. In NP-complete problem classes, the phase transition in solubility usually occurs over a similar range of κ [7].

Wallace and Freuder studied a number of constraint ordering heuristics for AC3, two of which we have shown to be proxies for the minimise κ_{ac} heuristic [23]. Nudel derived theoretically 8 constraint ordering heuristics, to maximise the detection of dead ends in forward checking [16]. These heuristics are similar to combinations of those of [23], selecting a constraint with minimum $m_v(1-p_c)$, where v is a future variable and c is incident on the current variable.

There appears to be scope for ordering heuristics within AC6. AC6 maintains a set, the *Waiting-List*, of unsupported values waiting to be propagated. Each element of *Waiting-List* is a pair (j, b), where b is a value removed from the domain of the jth variable. Associated with (j, b) is the set of values S_{jb} that are immediately supported by (j, b) and require new support. One possible heuristic may be to select (j, b) from *Waiting-List* such that $|S_{jb}|$ is maximised. At present we are unsure what decisions within AC6 will tend to minimise constrainedness, but we think this is worth further investigation.

10 Conclusions

We have shown that the same methodology used to study phase transition behaviour in NP-complete problems works with a polynomial problem class, establishing arc consistency in constraint satisfaction problems. The same measure for the constrainedness of an ensemble of problems that locates the phase transitions in random NP-complete problems identifies the location of a phase transition in establishing arc consistency. A complexity peak for the cost of the AC3 and AC6 algorithms is associated with this transition. Finite size scaling of this constrainedness parameter models both the scaling of the probability transition and of the search cost for AC3. This model of search cost agrees with the theoretical cubic worst case for AC3 on problems at the phase transition. This measure of constrainedness and proxies for it can then be used as the basis for constraint ordering heuristics that can reduce the number of checks and revisions performed by AC3.

What general lessons might be learnt from this study? First, we can identify and locate phase transitions in polynomial problem classes using the same constrainedness parameter developed to study NP-complete problems. Second, problems at the phase transition in polynomial problem classes can be more difficult to solve than over- or under-constrained problems away from the phase boundary. Indeed, our empirical model of median search cost for problems at the phase transition agrees with the theoretical worst case. Problems from the phase transition might therefore be useful for suggesting worst case asymptotes or, as in NP-complete domains, for benchmarking competing algorithms. Third, algorithms for polynomial problems can benefit from heuristics to reduce search. As with NP-complete problems, minimising constrainedness (and proxies for it which are cheap to compute) may provide the basis of useful heuristics. Finally, other polynomial problems (for example, path consistency, Horn satisfiability and polynomial approximation procedures for NP-complete problems) as well as other complexity classes might benefit from similar phase transition analysis.

References

1. Michael N. Barber. Finite-size scaling. In *Phase Transitions and Critical Phenomena, Volume 8*, pages 145–266. Academic Press, 1983.
2. C. Bessière. Arc-consistency and arc-consistency again. *Artificial Intelligence*, 65:179–190, 1994.
3. B. Bollobas. *Random Graphs*. Academic Press, 1985.
4. P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proceedings of IJCAI-91*, pages 331–337, 1991.
5. V. Chvatal and B. Reed. Mick gets some (the odds are on his side). In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 620–627. IEEE, 1992.
6. I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. Scaling effects in the CSP phase transition. In *Principles and Practice of Constraint Programming (CP-95)*, pages 70–87. Springer, 1995.
7. I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The constrainedness of search. In *Proceedings of AAAI-96*, pages 246–252, 1996.
8. I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The scaling of search cost. In *Proceedings of AAAI-97*, 1997.
9. I.P. Gent and T. Walsh. Phase transitions and annealed theories: Number partitioning as a case study. In *Proceedings of ECAI-96*, pages 170–174, 1996.
10. S.A. Grant and B.M. Smith. The arc and path consistency phase transitions. Report 96.09, Research Report Series, School of Computer Studies, University of Leeds, March 1996.
11. S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264:1297–1301, May 27 1994.
12. A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.
13. A.K. Mackworth and E.C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25:65–74, 1985.
14. D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of AAAI-92*, pages 459–465. AAAI Press/The MIT Press, 1992.
15. R. Mohr and T.C. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28:225–233, 1986.
16. B. Nudel. Consistent-labeling problems and their algorithms: Expected-complexities and theory-based heuristics. *Artificial Intelligence*, 21:135–178, 1983.
17. D. Sabin and E.C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *Proceedings of ECAI-94*, pages 125–129, 1994.
18. T. Schiex, J-C. Régin, C. Gaspin and G. Verfaille. Lazy Arc Consistency. In *Proceedings of AAAI-96*, pages 216–221, 1996.
19. B. Selman and S. Kirkpatrick. Critical behavior in the computational cost of satisfiability testing. *Artificial Intelligence*, 81:273–295, 1996.
20. B.M. Smith and M.E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81:155–181, 1996.
21. E.P.K. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1993.
22. R.J. Wallace. Why AC-3 is almost always better than AC-4 for establishing arc consistency in CSPs. In *Proceedings of IJCAI-93*, pages 239–245, 1993.
23. R.J. Wallace and E.C. Freuder. Ordering heuristics for arc consistency algorithms. In *Proc. Ninth Canad. Conf. on AI*, pages 163–169, 1992.