Phase Transitions in Classical Planning: an Experimental Study

Jussi Rintanen
Albert-Ludwigs-Universität Freiburg
Institut für Informatik, Georges-Köhler-Allee
79110 Freiburg im Breisgau
Germany

Abstract

Phase transitions in the solubility of problem instances are known in many types of computational problems relevant for artificial intelligence, most notably for the satisfiability problem of the classical propositional logic. However, phase transitions in classical planning have received far less attention. Bylander has investigated phase transitions theoretically as well as experimentally by using simplified planning algorithms, and shown that most of the soluble problems can be solved by a naïve hill-climbing algorithm. Because of the simplicity of his algorithms he did not investigate hard problems on the phase transition region. In this paper, we address exactly this problem.

We introduce two new models of problem instances, one eliminating the most trivially insoluble instances from Bylander's model, and the other restricting the class of problem instances further. Then we perform experiments on the behavior of different types of planning algorithms on hard problems from the phase transition region, showing that a planner based on general-purpose satisfiability algorithms outperforms two planners based on heuristic local search.

Introduction

The existence of phase transitions in many types of problems in artificial intelligence is well-known since the papers by Huberman and Hogg [1987] and Cheeseman, Kanefsky and Taylor [1991]. A detailed investigation of phase transitions in the satisfiability problem of the classical propositional logic was carried out by Mitchell, Selman and Levesque [1996]. Their space of problem instances (for a fixed number of propositional variables) consists of all sets of 3-literal clauses. In this space certain phase transition phenomena have been found both empirically and analytically: as the ratio of number of clauses and number of propositions approaches 4.2 from below, the probability that the formula is satisfiable increases. Similarly, when the ratio approaches 4.3 from above, the probability that the formula is satisfiable decreases. At about ratio 4.27 the probability is 0.5, far below 4.27 the probability is 1, and far above it is 0.

The phase transition from 1 to 0 at 4.27 coincides with the difficulty of testing the satisfiability of the formula: all known algorithms take exponential time in the size of the formulas when they have clauses to propositions ratio 4.27, and on many good algorithms the runtimes decrease sharply when going in either direction from 4.27. This is the *easy-hard-easy* pattern at the phase transition region.

Similar phase transitions and easy-hard-easy patterns have been discovered in many difficult computational problems, including classical planning. Bylander [1996] carries out an investigation on phase transitions in classical planning. He shows that in his model of sampling the space of problem instances, increasing the number of operators changes the problem instances from almost certainly not-having-a-plan to almost certainly having-a-plan.

Bylander further shows that almost all of the problem instances that are not too close to the phase transition region can be solved very efficiently with very simple planning algorithms. Inexistence of plans can in easy cases be tested with an algorithm that tests for a simple syntactic property of the problem instances. Similarly, plans for problem instances with a high number of operators can be found by a simple one-shot hill-climbing algorithm that does not do any search. But, unlike in the present work, Bylander does not carry out an empirical investigation of the actual computational difficulty of more realistic planning algorithms in the phase transition region: his algorithms, which he shows to be very effective outside the phase transition region, do not solve problems in the phase transition region.

Phase transitions in classical planning are closely related to the properties of random graphs [Bollobás, 1985]. The classical planning problem is the s-t-reachability problem in the transition graph encoded by the problem instance. As shown for random graphs, as the probability of edges between nodes increases, at certain probability a giant component, a set of nodes with a path between any two nodes, emerges, consisting of most of the nodes in the graph. This corresponds to having a set of operators with which there is a plan from almost any initial state to almost any goal state.

Along with similarities to random graphs, there are also important differences. The first difference is that unlike in most work on random graphs, the transition graphs in planning are directed. Second, the succinct representation of the transition graph induces a neighborhood structure not present in random graphs: for example, if the number of state variables changed by any operator is bounded by n, there are never any edges from a state to another state that differs in more than n state variables. Therefore results about random graphs are not directly applicable to analyzing properties of succinctly represented planning problems.

In this paper we complement Bylander's pioneering work on phase transitions in planning. Bylander's analysis focused exclusively on easy problem instances outside the phase transition region. We empirically investigate difficult problem instances inside the phase transition region. We also propose an improvement to Bylander's method for sampling the space of problem instances, well as propose a new model with the requirement that every state variable occurs the same number of time in an operator effect.

Random Sampling of Problem Instances

In this section we discuss the model of randomly sampled problem instances proposed by Bylander [1996], which we will call Model B, and two refinements of this model, called Model C and Model A. Each model is parameterized by parameters characterizing the size of problem instances in terms of the number n of state variables and the number m of operators, as well as properties of the operators, like the number s of literals in the precondition and number t of literals in the effect. Further, there are parameters for the description of the goal states.

Every combination of parameters represents a finite class of problem instances. Even though these classes are finite, the number of instances in them are astronomic for even relatively small parameter values, and the way they are investigated is by randomly taking samples from them, testing their computational properties, and then drawing more general conclusions about the members of the class in general.

Our interest in these classes of problem instances is to try to conclude something about their *computational difficulty* on the basis of the parameter values describing them. For example, our experiments suggest that the computational difficulty of the problem instances in Model A – for all the planners experimented with – peaks when the ratio between the number m of operators and the number n of state variables is about 2 (assuming certain fixed values for the rest of the parameters.)

This approach allows us to generate an unbounded number of problem instances, most of which are difficult, and these instances can be used in experimenting with different kinds of planning algorithms, and concluding something about the properties of these algorithms with respect to *most* of the instances having certain properties.

Next we define the models of problem instances, and after that continue by presenting the results of experiments performed with different types of planning algorithms.

Model B (Bylander)

Bylander [1996] proposes two models for sampling the space of problem instances of deterministic planning, the *variable model*, in which the number of preconditions and effects vary, and the *fixed model*, with a constant number of preconditions and effects. These models are analogous to the constant probability model and the fixed clause length model for propositional satisfiability [Selman, Mitchell, & Levesque, 1996]. In this paper we consider the fixed model only. As shown by Bylander [1994], deterministic planning with STRIPS operators having two preconditions and two effects is PSPACE-complete, just like the general planning problem, but the cases with 2 preconditions and 1 effect as well as 1 precondition and 2 effects are easier. More than 2 preconditions or effects can be reduced to the case with 2 preconditions and 2 effects, and therefore both the fixed and the variable model cover all the problem instances in propositional planning.

Definition 1 (Model B) *Let n, m, s, g and g' be positive integers such that $s \leq n$, $t \leq n$ and $g' \leq g \leq n$. The class $\mathcal{C}^B_{n,m,s,t,g,g'}$ consists of all problem instances $\langle P, O, I, G \rangle$ such that*

1. *P is a set of n Boolean state variables,*
2. *O is a set of m operators $\langle p, e \rangle$, where*
   (a) *p is a conjunction of s literals, with at most one occurrence of any state variable,*
   (b) *e is a conjunction of t literals, with at most one occurrence of any state variable,*
3. *$I : P \rightarrow \{0, 1\}$ is an initial state (an assignment of truth-values to the state variables)*
4. *$G : P \rightarrow \{0, 1\}$ describes the goal states. It is a partial assignment of truth-values to the state variables: of the $|P|$ state variables g are assigned a value, and g' of them have a value differing from the value in the initial state I.*

In our experiments we consider the case with $s = 3$ preconditions and $t = 2$ effects only. This appears to provide more challenging problems than the class with only 2 preconditions.[1] Also, we only consider the goal state descriptions that describe exactly one state, and the values of all state variables in this state are different from their values in the initial state. The classes of problem instances we consider are hence $\mathcal{C}^B_{n,m,3,2,n,n}$ for different n and m.

Instead of using n and m to characterize the number of state variables and operators, we may use n and $cn = m$ for some real number $c > 0$ instead. The use of the ratio $c = \frac{m}{n}$ is more convenient when talking about problem instances of different sizes.

We hoped that Bylander's model would yield a phase transition for a fixed c that is independent of the number of state variables n, but this turned out not to be the case. The problem is that on bigger problem instances and any fixed c, the probability that at least one of the goal literals is not an effect of any operator goes to 1, and the probability of plan existence simultaneously goes to 0.

This can be shown as follows. Essentially, we have to choose $2cn$ operator effects from $2n$ literals, 2 effects for each of the cn operators. All ways of choosing operator effects correspond to all total functions from a $2cn$ element set to an $2n$ element set. One or more of the goal literals are not made true by any operator if the corresponding function

[1] Even though the case with 3 preconditions can be reduced to the case with 2 preconditions, this type of reductions introduce dependencies between state variables. As a result, there is no one to one match between instances in $\mathcal{C}^B_{n,m,2,2,n,n}$ and $\mathcal{C}^B_{n,m,3,2,n,n}$, and also their computational properties differ.

is not a surjection. As n increases, almost no function is surjection. Let A and B be sets such that $m = |A|$ and $n = |B|$. The number of surjections from A to B is $n!S(m, n)$, where $S(m, n)$ is the number of partitions of an m element set into n non-empty parts, that is, the Stirling number of the second kind. We further get

$$n!S(m, n) = \sum_{k=0}^{n} (-1)^k \left(\begin{array}{c} n \\ k \end{array} \right) (n - k)^m.$$

What is the asymptotic proportion of surjections among all functions? We divide the number of surjections by the total number of functions from A to B, that is n^m, and get

$$\sum_{k=0}^{n} (-1)^k \left(\begin{array}{c} n \\ k \end{array} \right) \left(\frac{n-k}{n} \right)^m.$$

As n approaches ∞ (with $m = cn$), the limit of this expression for any constant c is 0. That is, as the sets increase in size, an infinitesimally small fraction of all functions are surjections.

This means that as the number n of state variables increases, there is no constant c so that cn operators suffices for keeping the probability of plan existence above 0, and an increasing number of operators is needed to keep the probability of having at least one operator making each state variable true high.

Even though the required ratio of operators to state variables increases only logarithmically (see [Bylander, 1996, Theorem 2]), we would almost characterize this as a *flaw* in Bylander's model of sampling the space of problem instances, especially because on bigger problem instances with few operators this is the dominating reason for inexistence of plans. See [Gent & Walsh, 1998] for a discussion of flaws in models for other computational problems.

Bylander intentionally includes these trivially insoluble instances in his analysis: his algorithm for insolubility detects exactly these instances, and no others.

Model C

We define a new model of random problem instances that does not have the most trivially insoluble problem instances Bylander's [1996] model has. To eliminate the most trivially insoluble instances we impose the further restriction that every literal occurs as an effect of at least one operator.

Definition 2 (Model C) *Let n, m, s, g and g' be positive integers such that $s \leq n$, $t \leq n$ and $g' \leq g \leq n$. The class $\mathcal{C}^C_{n,m,s,t,g,g'}$ consists of all problem instances $\langle P, O, I, G \rangle \in \mathcal{C}^B_{n,m,s,t,g,g'}$ in which for every $p \in P$, both p and $\neg p$ occur in the effect of at least one operator in O.*

This model eliminates all those problem instances that were recognized as insoluble by the algorithm Bylander [1996] used for plan inexistence tests. That some of the goal literals do not occur in any operator is a rather uninteresting reason for the inexistence of plans.

Model A

The way operators effects are chosen in Model B (the fixed model of Bylander [1996]) has a close resemblance to the fixed-clause-length model of sampling the space of 3CNF formula proposed by Mitchell et al. [1996]: randomly choose a fixed number of state variables and with probability 0.5 negate them. With propositional satisfiability this does not lead to any problems, but as we saw earlier, for operator effects it does. That a proposition does not occur in a clause does not mean that the clause would be more difficult to satisfy, but for a planning problem if a state variable does not occur as an effect this immediately means that certain (sub)goals are impossible to reach. Similarly, even when at least one occurrence is guaranteed, as in our Model C, when some of the state variables occur in the effects only a small number of times, on bigger problem instances similar phenomena often arise, like state variable A is made true only by an operator with B in the precondition, and B is made true only by an operator with A in the precondition. Because of this we think that a more interesting subclass of instances do not choose the effects independently like literals for clauses are chosen in the fixed-clause-length model for propositional satisfiability.

This leads to our Model A. The idea is that every state variable occurs as an operator effect (approximately) the same number of times, and the same number of times both positively and negatively. As a result, our Model A does not have some of the most trivially insoluble instances Model B and Model C have.

Definition 3 (Model A) *Let n, m, s, g and g' be positive integers such that $s \leq n$, $t \leq n$ and $g' \leq g \leq n$. The class $\mathcal{C}^A_{n,m,s,t,g,g'}$ consists of all problem instances $\langle P, O, I, G \rangle \in \mathcal{C}^B_{n,m,s,t,g,g'}$ in which for every $p \in P$, both p and $\neg p$ occur in the effect of either $\left\lceil \frac{tm}{2n} \right\rceil$ or $\left\lfloor \frac{tm}{2n} \right\rfloor$ operators.*

Experimental Analysis of the Phase Transition with Complete Algorithms

In this section we experimentally analyze the location of the insoluble-to-soluble phase transition under the models A and C of randomly generated problem instances, and evaluate different types of planning algorithms on hard problem instances from the phase transition region.

The planners are the following. First, the constraint-based planner we used does a translation to the propositional logic and finds plans by a satisfiability algorithm [Rintanen, Heljanko, & Niemelä, 2004]. We refer to this planner as SP. This is the satisfiability planning approach introduced by Kautz and Selman [1996]. SP finds plans of guaranteed minimal parallel length, because it sequentially tries every possible parallel plan length n and shows the inexistence of plans of that length before continuing with length $n + 1$. The shortest parallel plan, having length n, does not necessarily correspond to the shortest sequential plan, as there might be a parallel plan of length $n + 1$ or higher consisting of a smaller number of operators. The SAT solver we used was Siege version 3 [Ryan, 2003]. This solver is based on sophisticated clause-learning like other recent efficient SAT solvers, including zChaff. The runtimes we report are the sums of solution times reported by Siege, and do not include the time spent by our front-end that produces the formulae.

Second, from the recently popular family of planning algorithms based on distance heuristics and local search [Bonet & Geffner, 2001] we have the FF planner [Hoffmann & Nebel, 2001].

Third, we have the LPG planner by Gerevini and Serina [2002], which is based on similar ideas to satisfiability planning, namely the planning graph of the GraphPlan algorithm [Blum & Furst, 1997], but uses local search and does not increase plan length sequentially. LPG and FF were chosen because of their good performance on the standard benchmark sets; they can be considered to represent the state of the art with respect to these benchmark sets.

In the first part of our investigation, we produced a large collection of soluble problem instances with 20 state variables and tested the runtime behavior of the three planners on them. The plan inexistence tests were carried out by a complete BDD-based planner that traverses the state space breadth first and finds shortest existing plans or reports that no plans exist. Runtime of this planner is proportional to the number of reachable states, and it solved every problem instance in about two minutes or less. The other planners do not have a general effective test for plan inexistence.

In the second part of the investigation, described in the next section, we produced bigger problem instances with 40 and 60 state variables. Because only one of the planners was sufficiently efficient to solve a large fraction of bigger instances, we restricted the investigation of these bigger problem instances to this planner. For these bigger problem instances we could not perform complete solubility tests: the runtimes of our BDD-based planner are too high when the number of state variables becomes higher than 20.

For model A, we produced between 350 and 608 soluble problem instances for each ratio of operators to state variables, and for model C between 89 and 784. For smaller ratios this involved testing the solubility of up to 50000 (for model A) and up to 20000 (for model C) problem instances with the BDD-based planner with a complete solubility test. Restrictions on available CPU resources prevented us from finding still more soluble instances.

The diagrams in Figure 1 depict the empirically determined phase transition in planning with 20 state variables as well as corresponding runtimes on the three planners. The times are on a 3.6 GHz Intel Xeon processor with a 512 KB internal cache.

The diagrams in Figure 2 depict the average plan lengths on the three planners on problem instances with 20 state variables. The diagram in Figure 3 depicts the plan length (number of operators) and the number of time steps (parallel length) in the parallel plans produced by the SP planner.

Discussion of the Results on 20 State Variables

In this section we discuss the solubility and runtime data shown in the diagrams.

The Phase Transition The phase transition curve depicted in Figure 1 matches the expectation of how problem instances turn from almost certainly insoluble to almost certainly soluble as the number of operators grows. The change from insoluble to soluble in Model C is not very abrupt. Significant numbers of soluble instances emerge at operators-to-variables ratio 2 (earliest possible is 1 because a smaller number of operators with only two effects makes it impossible to make all of the goal literals true), and almost certain solubility is reached slowly, and even at ratio 9 solubility is still only about 0.99 and still growing slowly. In Model A, on the other hand, the transition is steeper, and solubility with probability 1.0 is reached soon after ratio 6. We think that this difference is due to the presence of state variables occurring as an effect only very few times in Model C, which is the difference to Model A.

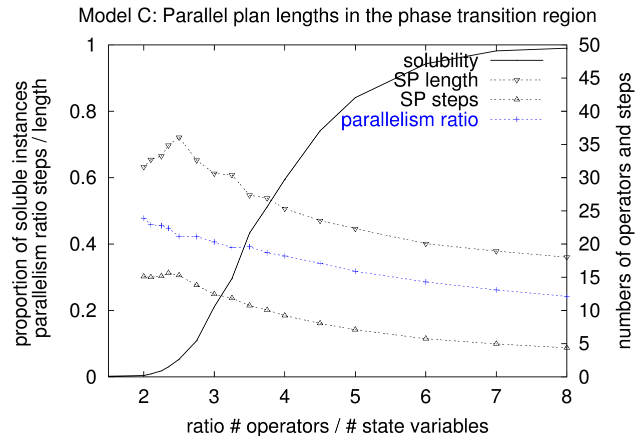

Figure 3: Number of operators (length) and time steps (parallel length) in plans found by SP for problem instances with 20 state variables

The Easy-Hard-Easy Pattern and the Planner Runtimes In Figure 1, there is a transition from hard to easy problem instances as the ratio c of operators to state variables grows beyond 3. That there is a transition from easy to hard instances at the left end of the curves before ratio 3 is less clear. Below ratio 2 there were no soluble instances, and we do not have any data on runtimes on insolubility testing because none of the planners has a general and effective insolubility test. Presumably, determining insolubility on most instances with very few operators is computationally easy.

It seems that difficulty in Model A peaks at about operators-to-variables ratio 2, and in Model C at about ratio 2.5, at least for SP and when looking also at the data on bigger instances in Figure 6. FF's runtime curve does not suggest the same, as the curves peak later respectively at about ratios 2.7 and 3. However, the curves for the number of instances FF and LPG did not solve within the time limit of 10 minutes (depicted in Figure 4) also suggest that these ratios 2 for Model A and 2.5 for Model C are the most difficult ones. Possibly for FF at slightly later ratios there are several difficult instances that are solved within the time limit but have a high runtime that contributes to the peak in the runtime curve.

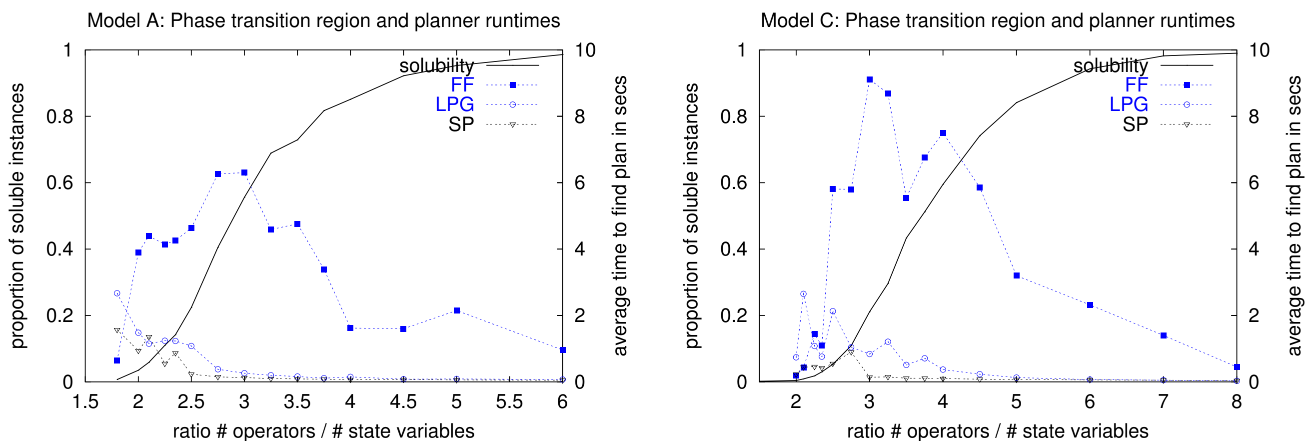

Figure 1: The plan existence phase transition and the passage from hard to easy in planner runtimes on soluble problem instances with 20 state variables. The averages do not include a number of problem instances that were not solved in 10 minutes.

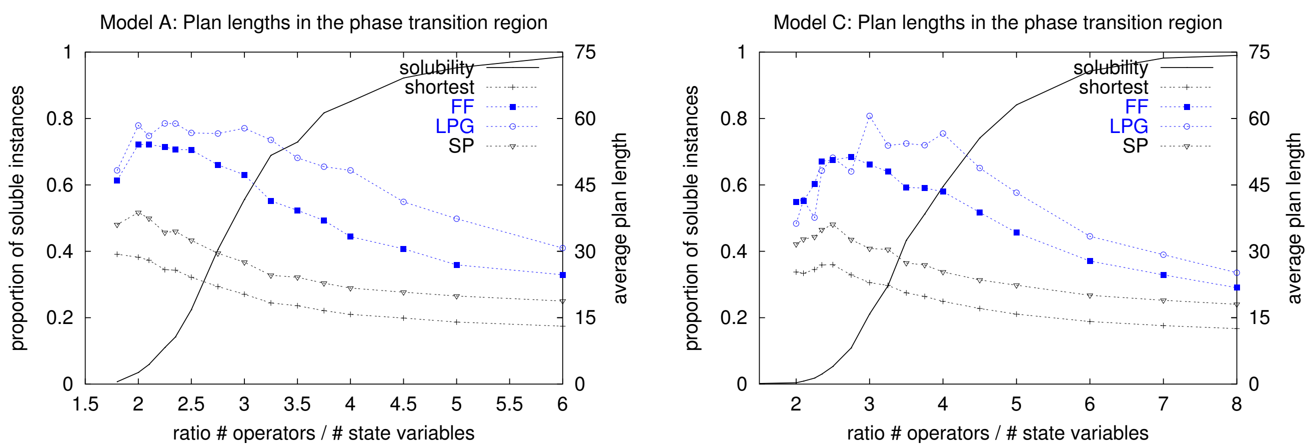

Figure 2: Average shortest plan lengths in the phase transition region for problem instances with 20 state variables, and the average lengths of the plans found by the three planners.

A difference to satisfiability in the propositional logic is that in both of the models the peak in difficulty does *not* appear to be near the 50 per cent solubility point.

The SP runtimes are lower than the LPG and FF runtimes. For model C there were few instances with a runtime over one minute, and the highest runtime of any of the Model C instances was 275 seconds. Model A appears to be more difficult, with two instances exceeding the time limit of 10 minutes (runtimes 1460 seconds and 870 seconds) at ratio 2.1. Most problem instances are solved in a fraction of a second, with medians between 0.05 and 0.20 seconds for all ratios in both models A and C.

LPG solves many instances almost immediately, but a high percentage of problems (about 10 between ratios 1.8 and 2.25 for Model A and between ratios 2.0 and 2.35 for Model C) are not solved in ten minutes, and a smaller percentage until the end of the phase transition region (6.5 for Model A and 8 for Model C). Because these are not included in the curves, average LPG runtimes are higher than those depicted in the figure. Also FF runtimes vary a lot. Most instances are solved quickly, but many instances are solved barely below the time bound of 600 seconds, and many instances (5 for Model A and about 25 for Model C) are not solved under 600 seconds and are not included in the averages. For Model A these are between ratios 2.5 and 3.75 and for Model C between 3 and 5. Figure 4 depicts the proportion of soluble 20 variable instances in model A that remained unsolved by LPG and FF in ten minutes. For FF we also give the curve depicting the 10 minute success rate on soluble instances with 40 variables. With 20 state variables FF's success rate is close to 100 per cent but with 40 state variables it is 4.3 per cent on the hardest instances and still only about 90 percent at the very easy ratio of 6.

The distribution of runtimes on the planners have what is known as a heavy tail [Gomes *et al.*, 2000], that is, the runtimes do not concentrate around the average, and there is a substantial number of instances with a runtime well above the average. Even though LPG uses fast restarts – which was

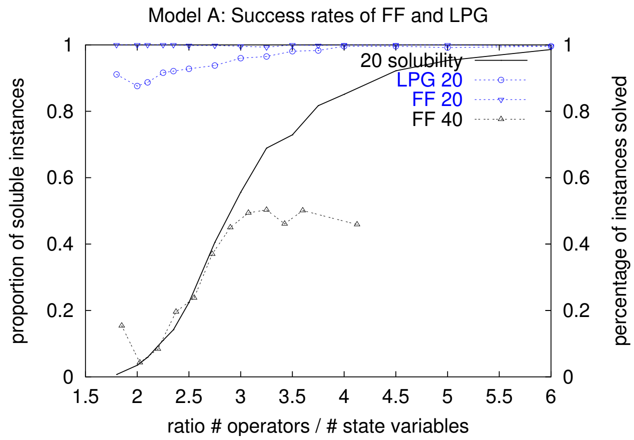

Figure 4: Percentage of soluble problem instances with 20 state variables FF and LPG solved in 10 minutes. For FF and 40 state variables the curve depicts the proportion of the number of instances solved by FF to the number of instances solved by SP. The FF success rate on 40 state variables is an upper bound because SP very likely missed some of the soluble instances.

proposed by Gomez et al. as a technique for weakening the heavy tail of runtime distributions on single instances – and FF does not, FF fares better than LPG.

Using the average as a characterization of runtimes when the distribution of runtimes has a heavy tail, as it has in our case, is not always meaningful [Gomes *et al.*, 2000], but we still decided to use it instead of the median because most of the 20 state variable instances are very easy to solve and the median completely ignores the difficult instances that distinguish the planners on problem instances of this size. The problem with heavy-tailed distributions is that there is a dependency on sample size: in general, the more samples are taken the higher will the average be because of an increased likelihood of obtaining extremely difficult instances. To obtain smooth runtime curves we should have tested a far more higher number of problem instances.

Plan Lengths Plan lengths in Figure 2 follow an interesting pattern. The lengths of shortest existing plans peak at the left end of the curve, followed by a slow decline. One would expect that the instances with short plans were those with many operators because there is more choice for choosing operators leading to short plans, and this is indeed the case. As the number of operators is increased, the asymptotic length of shortest plans will be 0.5 times the number of state variables, because there will be with a very high probability a sequence of operators that each make two of the goal literals true and are applicable starting from the initial state. Indeed, with the 20 state variable problems in Model C, at ratio 9 the shortest plans on average contain only 12 operators or 0.6 times the number of state variables.

The plan lengths for the different planners, none of which is guaranteed to produce the shortest plans, more or less follow the pattern of shortest plans, with the exception that the plan length does not peak at the left end but slightly later. The constraint-based planner SP produces plans that are relatively close to shortest ones (average lengths between 1.27 and 1.37 times the shortest on the difficult problems, and about 1.40 times on the easiest in Model C). This is because the guarantee that shortest parallel plans are found implies that the number of operators in the plans cannot be very high.

FF's plans are often about twice the optimal for the most difficult problem instances, and LPG's plans often three times the optimal. LPG plan lengths in Model C appear to peak at a higher operators-to-variables ratio than SP and FF.

The relations between the plan length and the number of time steps in the plans produced by SP, depicted in Figure 3, are what one would expect: with easier problems with more operators there are many plans to choose from, with a high number of plans with a small number of time points. In the more constrained problems there is on average 2 operators per time point, and this increases to almost 5 for the easiest problems.

Experimental Analysis of the Phase Transition on Higher Number of State Variables

Because the ability of FF and LPG to find plans on hard problem instances declines quickly as the number of state variables exceeds 20, the experiments with 40 and 60 state variables were made with SP only. Because SP does not have an insolubility test, we considered those problem instances insoluble for which we had not found plans during a 10 minute evaluation of formulae for plans lengths up to a fixed upper bound length. So we may have missed plans because of the timeout, or because we did not consider sufficiently long plans. However, because our plan length upper bound was substantially higher than the average lengths of plans actually found, the plan length restriction would seem to be the smaller source of missed soluble instances.

Phase transition for bigger problem instances is depicted in Figure 5. The solubility test can fail in one direction: if no plan was found, it could be because it was longer than what we tried or we had to terminate the run because of high runtime, and hence the actual solubility curve may be higher than the one we were able to determine with SP.

Even taking into account the one-sided error in detecting solubility, it is obvious that the phase transition region becomes narrower and the change from insoluble to soluble becomes steeper as the number of state variables increases. It is clear especially from the curve for Model A that the probability 1 solubility is reached earlier on problem instances with more state variables. The curves are also compatible with the idea that SP is capable of solving a large fraction of the bigger difficult soluble instances with 60 state variables, but to show that this is indeed the case we would need efficient algorithms for determining insolubility.

Average runtimes of solved instances are given in Figure 6. The median runtimes of solved instances are given in Figure 7. The heavy-tailed character of the distribution of runtimes becomes clear in the runtime curves. On some ratios the presence of a small number of very difficult instances causes the curve to peak so that the average runtime

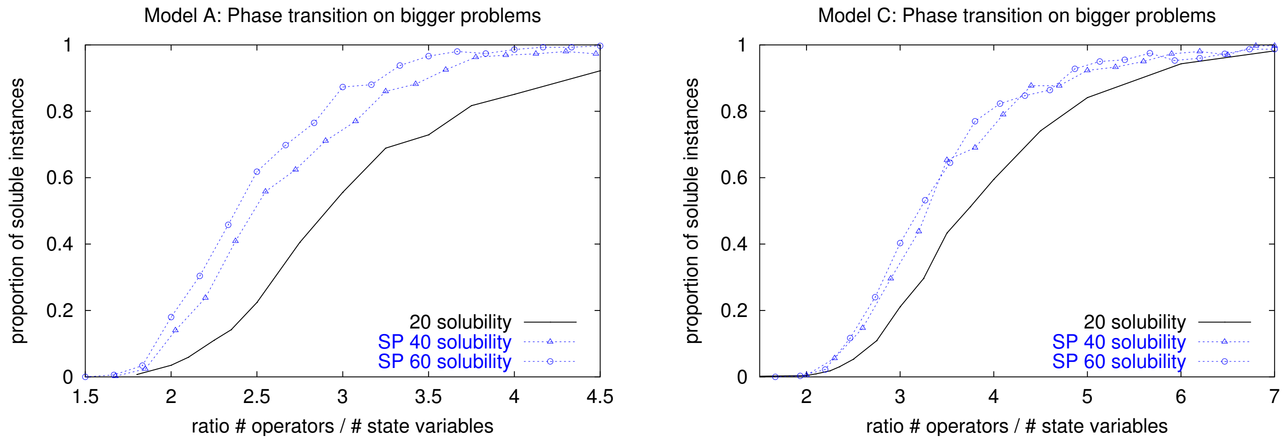

Figure 5: Phase transition on problem instances with 20, 40 and 60 state variables, as determined by the SP planner. Any problem instance that was not solved under 10 minutes or was shown not to have plans of a given maximum length was considered insoluble.

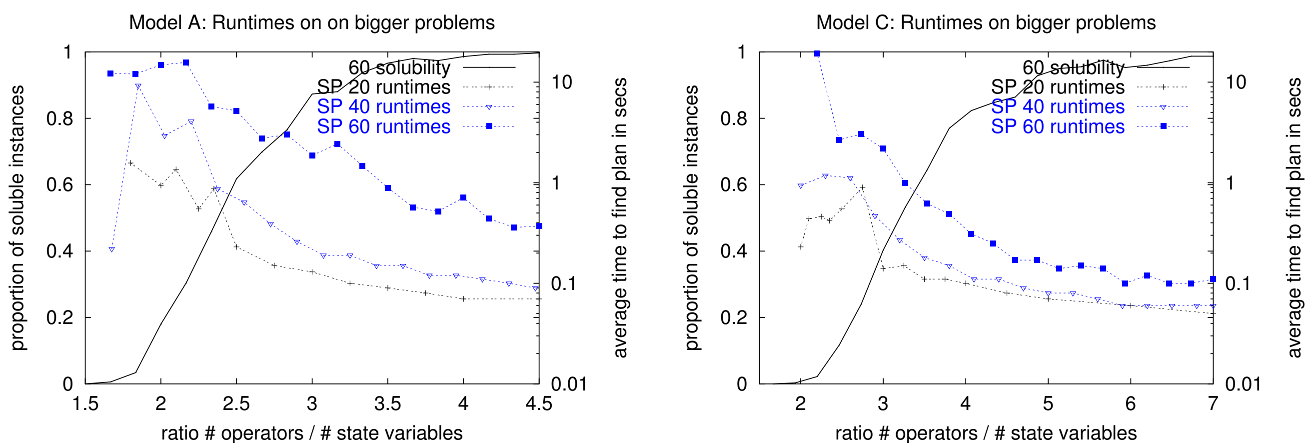

Figure 6: Average SP runtimes on problem instances with 20, 40 and 60 state variables

on 20 state variables appears on some ratios to be very close or higher than that of 40 state variables, and similarly for the curves for 40 and 60 state variables. The curves are not smooth because we only solved a moderate number of instances for each ratio (between 500 and 300, depending on the ratio), and for the smaller ratios the number of soluble instances is small.

The average plan lengths are depicted in Figure 8. The lengths grow slightly faster than the increase in state variables.

Discussion of the Results

The standard experimental methodology in planning is the use of problem scenarios resembling potential real-world applications, like simplified forms of transportation planning, simple forms of scheduling, and simplified control problems resembling those showing up in autonomous robotics and other similar areas. For details see [McDermott, 2000; Fox & Long, 2003]. There does not appear to be an attempt to identify inherently difficult problems and problem instances. In fact, many of the standard benchmarks are solvable in low polynomial time by simple problem-specific algorithms, and hence are computationally rather easy to solve. We believe that these properties of benchmarking strongly affects what kind of algorithms are considered good and bad.

Our empirical results on the computational behavior of the algorithms complement those obtained from the standard benchmarks. Planners based on heuristic local search have been very popular in recent years, mainly because of their success in solving the standard benchmark sets. Our results suggest that heuristic local search might be far weaker on difficult problems that differ from the standard benchmark problems.

Why do the heuristic search planners fare worse than satisfiability planning on the random problem instances from the phase transition region, and why do they fare relatively much better on the standard benchmarks?

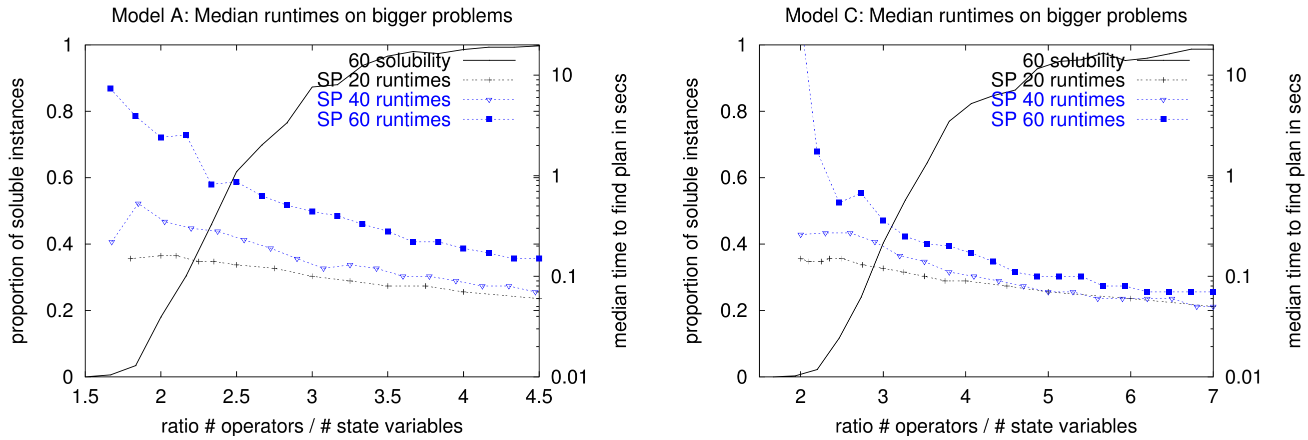

Figure 7: Median SP runtimes on problem instances with 20, 40 and 60 state variables

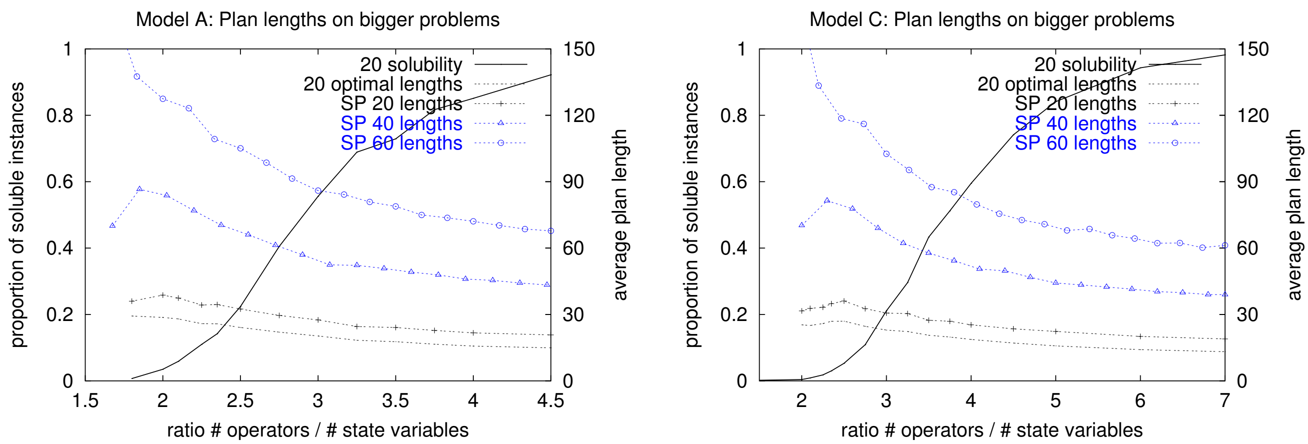

Figure 8: SP plan lengths on problem instances with 20, 40 and 60 state variables

FF is based on heuristic local search in the state space. The main reason for the recent popularity of heuristic local search was the discovery that polynomial-time computable domain-independent distance heuristics make some of the standard benchmarks much easier to solve [Bonet & Geffner, 2001]. Further improvements on these benchmarks have been obtained by ad hoc techniques inspired by some of the benchmarks themselves [Hoffmann & Nebel, 2001] but these techniques do not seem to address difficulties showing up in planning problems more generally. The weakest point in this class of planners is that when the distance heuristics fail to drive the search to a goal state quickly, there may be a huge state space to search for, and this takes place by explicitly enumerating the states. This makes these planners scale badly on difficult problems. What satisfiability planning has, and heuristic planners traversing the state space do not, is the ability to reason about the values of individual state variables at different time points. For this reasoning satisfiability algorithms use effective general-purpose techniques, like Boolean constraint propagation and clause learning. This way the problem representation in the propositional logic allows to make inferences about whole classes of plans and states, which are represented by partial assignments of truth-values to propositions [Rintanen, 1998]. For example, during plan search it is often inferred that there exist no plans with a state variable having certain value at a certain time point. This is possible without having to explicitly enumerate parts of the state space to test the reachability of all such states from the initial state. Presumably, this kind of reasoning greatly helps in solving inherently difficult problems with complicated operator interactions.

LPG's problem representation shares some of the properties the representation of planning as a satisfiability problem has, but LPG does not utilize the properties of the representation in the same extent general-purpose satisfiability algorithms do. For example, LPG uses forms of constraint propagation (no-op propagation [Gerevini & Serina, 2002]), but only in a restricted way.

An important question about the implications of the results of the present paper to planning more generally is how much similarities are there between difficult planning problems arising from practical applications and the difficult problems randomly sampled from the space of all problem instances. Practically relevant difficult problems often have surface structure quite different from the randomly sampled problem instances, but many techniques developed for planning, like symmetry reduction [Rintanen, 2003], can be employed in eliminating this surface structure to yield a less structured core of the problem instance. Similarly, many of the techniques used in satisfiability algorithms, like Boolean constraint propagation, attempt to get past the surface structure. What remains is a hard search problem without further structural properties to take advantage of. These unstructured problems may be close to randomly generated problem instances.

Notice that many algorithms specifically designed to solve problems randomly sampled from the space of all problem instances, like the survey propagation algorithm [Mézard, Parisi, & Zecchina, 2002] and similar local search algorithms for propositional satisfiability [Seitz & Orponen, 2003], are very weak in solving instances from practically more interesting problem classes. Also, more conventional satisfiability algorithms can be specialized for solving hard random problem instances [Dubois & Dequen, 2001]. However, the SP planner [Rintanen, Heljanko, & Niemelä, 2004], solves standard planning benchmarks with an efficiency that is comparable to – and in some cases exceeds that of – planners developed this kinds of benchmark problems in mind.

Conclusions and Related Work

In addition to the study by Bylander [1996], one of the few works directly related to phase transitions planning is by Slaney and Thiébaux [1998]. They investigate relationships between the difficulty of optimization and the corresponding decision problems. As an example they use the traveling salesman problem and blocks world planning, both of which can be represented in the framework of classical planning. Our work concentrated on the problem of finding an arbitrary plan. The corresponding optimization problem of finding the shortest or cheapest plan and the decision problem of finding a plan of at most a given cost would be more relevant from the perspective of many applications.

An important open problem is the analytic derivation of tight upper and lower bounds for the phase transition region. As suggested by research on the propositional satisfiability phase transition and our experiments, the phase transition region becomes increasingly narrow as the number of state variables increases. Analogously to the SAT phase transition, there is presumably an asymptotic phase transition point, where problem instances turn almost instantaneously from insoluble to soluble as the number of operators is increased. The techniques that easily yield upper bounds for the propositional satisfiability phase transition are not directly applicable to planning. The plan existence problem has a decidedly graph-theoretic character that separates it from the satisfiability problem of the propositional logic. The upper bounds derived by Bylander are applicable to Model C (when $n = g = g'$), but the lower bounds are not, and further, the bounds Bylander has derived are loose.

The experimental evaluation should be complemented by an analysis of techniques for determining inexistence of plans, a topic not properly addressed in planning research. Some recently proposed approaches to the same problem outside planning are based on satisfiability testing and would appear to be the best candidates to try for planning as well [McMillan, 2003; Mneimneh & Sakallah, 2003].

References

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1-2):281–300.

Bollobás, B. 1985. *Random graphs*. Academic Press.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1-2):5–33.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1-2):165–204.

Bylander, T. 1996. A probabilistic analysis of propositional STRIPS planning. *Artificial Intelligence* 81(1-2):241–271.

Cheeseman, P.; Kanefsky, B.; and Taylor, W. M. 1991. Where the really hard problems are. In Mylopoulos, J., ed., *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, 331–337. Morgan Kaufmann Publishers.

Dubois, O., and Dequen, G. 2001. A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In Nebel, B., ed., *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, 248–253. Morgan Kaufmann Publishers.

Fox, M., and Long, D. 2003. The third international planning competition: results and analysis. *Journal of Artificial Intelligence Research* 20:1–59.

Gent, I., and Walsh, T. 1998. Beyond NP: the QSAT phase transition. Technical Report APES-05-1998, University of Strathclyde, Department of Computer Science.

Gerevini, A., and Serina, I. 2002. LPG: a planner based on local search for planning graphs with action costs. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems*, 13–22. AAAI Press.

Gomes, C. P.; Selman, B.; Crato, N.; and Kautz, H. 2000. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of Automated Reasoning* 24(1–2):67–100.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Huberman, B. A., and Hogg, T. 1987. Phase transitions in artificial intelligence systems. *Artificial Intelligence* 33(2):155–171.

Kautz, H., and Selman, B. 1996. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, 1194–1201. Menlo Park, California: AAAI Press.

McDermott, D. 2000. The 1998 AI planning systems competition. *AI Magazine* 21(2):35–55.

McMillan, K. L. 2003. Interpolation and SAT-based model checking. In Hunt Jr., W. A., and Somenzi, F., eds., *Proceedings of the 15th International Conference on Computer Aided Verification (CAV 2003)*, number 2725 in Lecture Notes in Computer Science, 1–13.

Mézard, M.; Parisi, G.; and Zecchina, R. 2002. Analytic and algorithmic solution of random satisfiability problems. *Science* 297:812–815.

Mneimneh, M., and Sakallah, K. 2003. Computing vertex eccentricity in exponentially large graphs: QBF formulation and solution. In Giunchiglia, E., and Tacchella, A., eds., *SAT 2003 - Theory and Applications of Satisfiability Testing*, number 2919 in Lecture Notes in Computer Science, 411–425.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2004. Parallel encodings of classical planning as satisfiability. Report 198, Albert-Ludwigs-Universität Freiburg, Institut für Informatik.

Rintanen, J. 1998. A planning algorithm not based on directional search. In Cohn, A. G.; Schubert, L. K.; and Shapiro, S. C., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR '98)*, 617–624. Morgan Kaufmann Publishers.

Rintanen, J. 2003. Symmetry reduction for SAT representations of transition systems. In Giunchiglia, E.; Muscettola, N.; and Nau, D., eds., *Proceedings of the Thirteenth International Conference on Planning and Scheduling*, 32–40. AAAI Press.

Ryan, L. 2003. Efficient algorithms for clause-learning SAT solvers. Masters thesis, Simon Fraser University.

Seitz, S., and Orponen, P. 2003. An efficient local search method for random 3-satisfiability. *Electronic Notes in Discrete Mathematics* 16(0).

Selman, B.; Mitchell, D. G.; and Levesque, H. 1996. Generating hard satisfiability problems. *Artificial Intelligence* 81(1-2):459–465.

Slaney, J., and Thiébaux, S. 1998. On the hardness of decision and optimisation problems. In Prade, H., ed., *Proceedings of the 13th European Conference on Artificial Intelligence*, 244–248. John Wiley & Sons.