Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Parametrized Families of Hard Planning Problems from Phase Transitions

Eleanor G. Rieffel, Davide Venturelli, Minh Do, Itay Hen and **Jeremy Frank**
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

There are two complementary ways to evaluate planning algorithms: performance on benchmark problems derived from real applications and analysis of performance on parametrized families of problems with known properties. Prior to this work, few means of generating parametrized families of hard planning problems were known. We generate hard planning problems from the solvable/unsolvable phase transition region of well-studied NP-complete problems that map naturally to navigation and scheduling, aspects common to many planning domains. We observe significant differences between state-of-the-art planners on these problem families, enabling us to gain insight into the relative strengths and weaknesses of these planners. Our results confirm exponential scaling of hardness with problem size, even at very small problem sizes. These families provide complementary test sets exhibiting properties not found in existing benchmarks.

Introduction

The construction of efficient general-purpose planners has been the holy grail for the planning community since the inception of the field. Benchmark problems enable the evaluation of progress toward this goal. Currently, most benchmark planning problems are designed by extracting solvable problems from real-world applications. This approach has the benefit of tuning algorithms toward the applications from which the problems are obtained. The drawbacks include: (1) It is hard to determine what makes a domain hard or easy for certain planners. (2) It is hard to predict the performance of a given planner on new domains with novel structure. (3) There are polynomial-time algorithms for certain domains (Helmert 2003; Hoffmann 2005). (4) Evaluation takes place only on problems known to be solvable, which does not reflect most real-world applications.

A complementary approach is to design parametrized families of planning problems that contain instances that can be shown to be intrinsically hard in the typical case. This approach has certain advantages: (1) It supports analysis as to which types of planning problems, and which aspects of these problems, are hard or easy for certain planners and why. (2) It is meaningful to say that even quite small problems, ones that can be solved quickly by many planners, are hard because they are part of a hard family. (3) Analysis of the behavior of planners on these small problems can highlight strengths and weaknesses of various planners and planning algorithms, give reasons for these strengths and weaknesses, and enable prediction of the performance of these algorithms on much larger instances. To date, however, parametrized families of hard planning problems have been hard to find, and few were known prior to this work.

Our aim is to establish parameterized sets of planning problems that (1) are intrinsically hard, and therefore challenging for all planners and types of planning algorithms; (2) are controllable by parameters that provide an exponential increase in problem hardness with linear size increase, and enable the establishment of small planning problems as hard instances (3) have explainable reasons for their hardness; (4) contain structures that exist in planning and scheduling domains of practical importance.

For many NP-complete problems, phase transitions from almost always solvable to almost always unsolvable have been observed, with the transition becoming sharper as the size of the problems increases. Hard problems tend to lie at the phase transition threshold (Cheeseman, Kanefsky, and Taylor 1991). Some NP-complete problems naturally lend themselves to planning problems. Here, we derive planning problems from well-studied graph-theoretic problems for which phase transition results are known, and for which the hardness of the problems at the threshold has been confirmed in empirical studies. The parametrized families of planning problems we designed fall into two classes: navigation-style families of planning problems derived from Hamiltonian path problems, and scheduling-style families derived from graph coloring problems. Many real-world applications of planning have aspects of both navigation and scheduling.

Our contributions include: (1) Benchmark sets of navigation and scheduling type planning problems derived from well-known combinatorial problems that exhibit phase transitions. (2) Hard problems of arbitrary size, including small instance involving only 10-40 grounded actions, two to three orders of magnitude smaller than typicial benchmark problems which involve thousands or tens of thousands of grounded actions. (2) Unsolvable as well as solvable problems. (3) Investigation of phase-transition in planning, using contemporary state-of-the-art planners. (4) Insights into the relative strengths and weaknesses of these planners.

Related work

Phase transitions in combinatorial optimization problems have long been recognized (Huberman and Hogg 1987), with hard problem instances generally found at these phase transitions (Cheeseman, Kanefsky, and Taylor 1991; Selman, Mitchell, and Levesque 1996). Bylander (Bylander 1996) investigated phase transitions in planning by generating random planning problems with the ratio of actions to state variables as the control parameter. He found that the probability of a instance having a plan was almost 0 for small ratios (too few actions) and almost 1 for large ratios (too many actions). In this pioneering work, Bylander used very simple algorithms, far from the sophisticated algorithms of current state-of-the-art planners, to show the easiness of determining solvability or unsolvability far from the phase-transition region. He did not investigate the computational difficulty at the phase transition for realistic planning algorithms. Also, his phase transition was not sharp because key differences between random graphs and state-space transition graphs in planning mean that his problems had more regularity than is ideal.

Slaney & Thiebaux (Slaney and Thiebaux 1998) investigated phase transitions for both optimization and decision problems in the popular BlocksWorld planning benchmark domain. Their empirical evaluation identified phase-transitions at certain ratios of the number of towers in the goal configuration to the total number of blocks. Their use of a search strategy customized to this domain means that it does not truly resemble algorithms employed by current state-of-the-art planners, and so the result is not conclusive.

Subsequently, Rintanen (Rintanen 2004) improved upon the early work by Bylander (Bylander 1996). His analysis focused on problems within the phase-transition region instead of in the easy regions. He also introduced two alternative models with additional restrictions to eliminate the most trivially unsolvable instances. Unlike the previous work, he used state-of-the-art planners for the empirical evaluation. The results show the easy-hard-easy behavior for the FF planner, but is not conclusive for the other two planners (LPG and SP). While the problems generated are better than those in Bylander's pioneering work, they do not represent identifiable aspects typical of real-world planning domains.

More recently, Rintanen (Rintanen 2012a) generated parametrized families of hard planning instances that differ from the previous work on several fronts. First, he generated only proven solvable instances. Second, instead of originating from random graphs, the hardness of this family is based on controlling the number of directed paths from the initial state to the goal states. Unlike our problems, his do not encapsulate well-recognized structures essential to many planning applications. Also, as shown in our results section, our problems are much harder than his at small instance sizes.

Porco et al. (Porco, Machado, and Bonet 2011) developed a tool for translating NP-complete problems into planning problems, specifically STRIPS fragments. While they applied their tool to the directed Hamiltonian path problem and the graph coloring problem, among others, they did not look specifically at problems in the phase transition region. Moreover, while they used the M planner to test the PDDL problems generated by the tool, we are interested in planner analysis and understanding the relative strengths and weaknesses of multiple planners on hard parameterized planning problems. Rintanen (Rintanen 2012b), extending the work of Porco et al., compares the performance of many planners on a large number of problems, including some problems translated from NP-complete problems, but not specifically in the phase transition region.

Parameterized Families of Navigation-Style Planning Problems

Navigation is a critical component in many planning applications and existing planning benchmarks (Long and Fox 2003; Helmert 2003; Hoffmann 2005). Rover navigation is one such domain. Given a list of locations a rover must visit to, say, take picture or analyze samples, the planner must find a route that makes optimal use of resources, such as time and power, satisfies multiple constraints, and achieves all goals. Under the assumptions that each location need be visited only once, the high-level navigation problem is similar to the Hamiltonian Path problems we investigate.

Planning problems from undirected Hamiltonian path (UHP)

The undirected Hamiltonian path (UHP) problem on an undirected graph $G(V, E)$, with n vertices V and a set of edges E, is to find a path that visits each node exactly once. A planning problem instance based on this graph may be formulated as follows. For each vertex v, there are:

- An action a_v representing visiting v.
- A 'goal' state variable s_v^g to indicate that v needs to be visited; $s_v^g = T$ (true) means v has been visited.
- An 'internal' state variable s_v^i represents whether or not v has been visited. Specifically, $s_v^i = T$ means v has *not* been visited while $s_v^i = F$ (false) means that it has been visited. This variable ensures that each vertex can be visited at most once. While including both s_v^g and s_v^i (which always have opposite values) seems redundant, it is necessary because the standard STRIPS planning representation allows only positive action preconditions and goals.
- An 'external' state variable s_v^e represents whether or not the vertex v can currently be visited given the edge structure of the graph. Specifically, it is set to T by an action $a_{v'}$ corresponding to visiting a vertex v' that is connected to v by an edge. Otherwise, it is set to F.

Each action a_v has 2 preconditions: (1) $s_v^i = T$, which indicates that this action has not been used in the plan already, and (2) $s_v^e = T$, indicating that this action can legally follow the previous action.

Each action a_v has $n + 1$ effects: (1) $s_v^g = T$, to indicate that v has been visited, (2) $s_v^i = F$, thus excluding a_v from appearing twice in the plan, (3) sets each of the $n-1$ external variables $s_{v'}^e$ for each of the other vertices v': if there is an edge from v to v' then $s^e_{v'} = T$, enabling $a_{v'}$ to follow a_v; if there is no edge from v to v' then a_v sets $s^e_{v'} = F$, preventing $a_{v'}$ from following a_v.

The initial state has all goal variables $s^g_v = F$ while all internal and external variables s^i_v and s^e_v have value T. Thus, any of the n actions a_v can be performed at the start. A valid plan is a sequence of the n actions that corresponds to a path along the edges that visits all vertices exactly once.

Problem generation: We randomly generate Erdös-Rényi graphs $G_{n,p}$ (Erdős and Rényi 1960), where n is the number of vertices and p is a connectivity parameter: for any pair of vertices, include the edge between them with probability p. We then obtain a parametrized family of UHP-based planning problems, parametrized by n and p, from these graphs as described in the preceding paragraphs. We wrote a simple C++ program to generate these problems, and to produce the PDDL representation (a domain and problem file pair) for each instance.

Parametrized families of scheduling-type planning problems

Many planning applications include scheduling aspects (Chien et al. 2012). Scheduling, which deals with assigning resources and time to tasks while taking into account constraints, is in itself an important problem. Certain classes of scheduling problems correspond to graph coloring. For example, a scheduling problem, with a set of tasks and constraints that any pair of tasks competing for the same resource cannot be assigned the same time-slot, can be phrased as a *vertex coloring*, a well-known NP-complete problem. Specifically, the chromatic number (the smallest number of colors needed) represents the smallest number of time-slots needed to complete a corresponding schedule, thus representing the minimum makespan.

Planning problems from Graph Coloring (GC)

Given an undirected graph $G = \{V, E\}$ with n vertices, the planning problem to color G with k colors is formulated as follows. For each vertex v there are:

- k actions a^c_v representing coloring v with color c.
- A goal variable s^g_v representing whether or not v has been colored at all.
- A state variable s^c_v representing whether or not v has been colored with the color c.

Let $C(v)$ be the set of neighboring vertices that are connected to v by an edge. For each action a^c_v, there are $|C(v)|+1$ preconditions: (1) $s^g_v = F$, which indicates that v is not already colored; and (2) for each $v_i \in C(v)$, $s^c_{v_i} = F$, guaranteeing that none of neighboring v_i are already colored with color c.

Each action a^c_v has two effects: $s^g_v = T$ and $s^c_v = T$.

In the initial state, none of the vertices are colored: $\forall v \in V : s^g_v = F$, and $s^c_v = F$. The goal state requires that all vertices are colored: $\forall v \in V : s^g_v = T$. A plan is a sequence of n actions, each of which colors a vertex v.

Problem generation: As for the Hamiltonian path based problems, we obtain a parametrized family of graph-coloring-based planning problems by randomly generating Erdös-Rényi graphs $G_{n,p}$ for a variety of values of n and p. However, instead of relying on our own program, we extended the graph generator program described in (Culberson, Beacham, and Papp 1995), which provides methods to generate different types of graph controlled by various parameters, to output PDDL files containing the specification of planning problems derived from these graphs.

Results and Discussion

In this section, we present the results and analysis of the performance of several state-of-the-art planners on the different planning domains described above. We specifically focus on the phase transitions of these domains, which is where the typical problems are expected to be the hardest to solve. The order parameters for the phase transitions studied here are the same as those of the original graph problems.

We first confirm the existence of a solvable/unsolvable phase transition in each of these families as the connectivity parameter p is varied, and check that the problems in this transition region are the ones that are most challenging to tackle, taking the most time to solve or to show unsolvable, with easy-to-solve and easy-to-show-unsolvable problems on either side. We used a complete planner, FF, in both cases, with each data point showing the median runtime of 50 random instances for the relevant parameters. These results were collected using a 64-bit RedHat Linux machine with 8 Intel Core I7 cores running at 2.4 Ghz with 8 GB of RAM.

We then did a thorough study, with hundreds of thousands of runs, examining the scaling behavior of the planners as problem size n increases and p is varied as a function of n to stay on the phase transition. We obtain this order parameter from the literature on the underlying problems. The exponential growth of the median runtime for all planners provides further evidence of the intrinsic difficulty of the problems at the phase transition. The absolute time and the slope of the exponential enable us to compare the efficacy of the planners on these problems. For these problems, we used the Ivy Bridge nodes of NASA's Pleiades supercomputer, Intel Xeon E5-2680v2 10-core processors running at 2.8 Ghz with 32 GB of RAM.

Planners used: To get representative results, we sought to use a set of planners that: (1) use different planning algorithms; and (2) are considered state-of-the-art (have strong performance on existing benchmarks). Specifically, we tested the following planners[1]: (1) FF (Hoffmann and Nebel 2001) and LAMA-2011 (Richter and Westphal 2010), representing the current dominating forward-state-space search framework; (2) LPG (Gerevini, Saetti, and Serina 2003) representing local search approach; and (3) M and Mp (Rintanen 2012b) representing the compilation approach.

We decided to exclude LAMA results from our analysis and graphs for a couple of reasons. First, unlike other

[1]We made small adjustments to each planner to output the information we need, especially for problems that are not solvable.

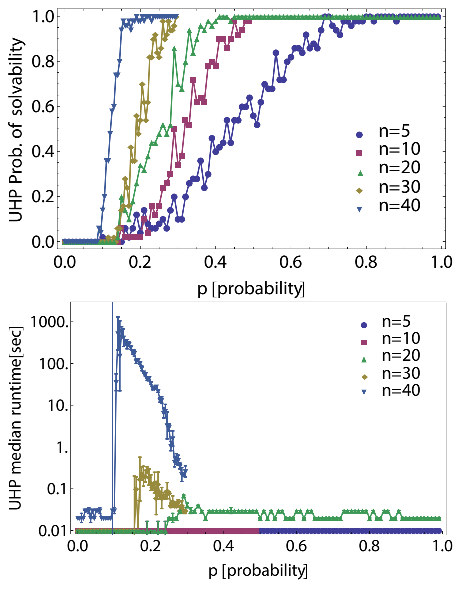


Figure 1: Navigation-type planning problems. **Top:** Fraction of solvable instances as a function of the connectivity parameter p for different problem sizes n. **Bottom:** Median runtime vs. p. The results shown are for the FF planner.

planners, LAMA has a preprocessing phase that can be costly. For the smallest problem in the UHP domain which other planners take a fraction of second to solve, LAMA took approximately 30 seconds for the pre-processing step alone. For the GraphColoring domain where the preprocessing time (≈ 0.07 sec) is tolerable, it typically solves only the smallest of these problems ($n = 8, 12, 16$) with the given time and memory resources, and even the worst performing planner on the scheduling type problems problems, FF, outperforms LAMA significantly on these cases.

Results on navigation-type planning problems

Figure 1 shows the median running time of the FF planner on the navigation-type planning problems inspired by undirected Hamiltonian path (UHP) problems. Figure 1 (Top) confirms the phase transition from unsolvable to solvable as the connectivity p is increased. The phase transition is sharp already at problem size $n = 40$, where n is the number of actions in the planning problem. While Figure 1 (Top) exhibits the phase transition, Figure 1 (Bottom) shows a significant difference in runtime between problems at the phase transition and those away from it, confirming that problems at the phase transition do pose the greatest challenge for the planners. The results shown are for FF, but we observe similar behavior in the other planners, M, Mp and LPG.

To compare different planners, we created test sets of 5000 problems at the phase transition, for each problem size from 16 to 40 at an increment of 2 (for a total of 65000 problems). We use the scaling parameter $p = (\log n + \log\log n)/n$ that has been established for the closely related Hamiltonian cycle problem (Komlós and Szemerédi 1983; Cheeseman, Kanefsky, and Taylor 1991).

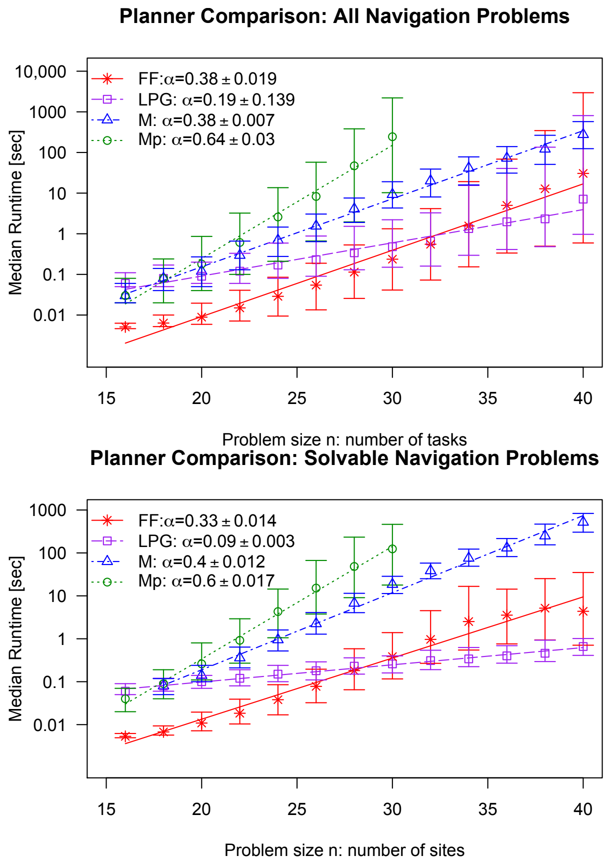


Figure 2: Median runtime vs. problem size at the phase transition, $p = (\log n + \log\log n)/n$, for navigation-type planning problems. The exponential scaling of runtime with problem size is evident for all planners. The exponential coefficient α is given in each case. (Top) All problems. (Bottom) Solvable problems.

Figure 2 (Top) shows the median runtime of the planners on problems at the phase transition. We use a two-hour (7200 sec) cutoff. Once the median runtime reaches the two-hour cutoff we no longer show results for that planner. The error bars are at the 35th and 65th percentile. For some of the points with high median runtimes, the error bars are cut off at 7200 seconds. The expected exponential scaling of difficulty, as measured by runtime, with the problem size is seen clearly. On typical problem instances, the FF and LPG planners outperform M and Mp. The relative slopes α suggest they will retain this advantage at larger problem sizes. At small problem sizes, FF performs best, but LPG overtakes FF for larger problem sizes. M significantly outperforms Mp. For the hardest problems, the error bars suggest that M is competitive with FF and LPG.

Figure 2 (Bottom) shows the runtime of different planners on the subset of solvable problems at the phase-transition. Upon restricting to solvable only instances, the relative performance of the planners does not change. Also, a clear exponential increase in solving time as problem size remains. The key difference is the significant reduction in the uncertainty of LPG's running time. This is expected because LPG uses a local search approach, which is incomplete and so performs particularly poorly on unsolvable instances.

Planners Comparison Analysis: Actions in the navigation-type planning domains are strongly "sequential" in the sense that: (1) each action of visiting a site C enables exactly the set of actions corresponding to visiting other sites that are connected to C by an edge; and (2) there are prevalent mutual-exclusion relations between actions: we cannot visit two sites in parallel. This type of constraint is known to put compilation-based planners such as M and Mp at a disadvantage (Kautz and Selman 1999). Because they need to bound the planning horizon h to create a SAT encoding, this type of domain may require M and Mp to go through multiple unsolvable encodings until it tries an h that is solvable. Moreover, when the problem is not solvable, it is also harder for M and Mp to discover that no matter how high the value of h, there is no solution. FF, on the other hand, can switch to a complete breadth-first-search algorithm that will exhaustively search until depth n to return the correct answer.

However, this type of domain is not a perfect fit for FF's heuristic either. Each solution is of the same length n, so when FF explores a search node X that is obtained via m actions from the initial state, all children of X will either have the same distance $n - m$ to the goals or are "dead ends" (cannot reach the goals). The equal heuristic values means FF cannot use them to differentiate between "good" and "bad" nodes to explore next, but the dead-end discovery will help it to eliminate many children nodes, especially for problems in the phase-transition region where there are few solutions and therefore many dead ends.

LPG's performance is similar to FF's because (1) it seeds its initial flawed plan to repair with FF's first relaxed plan; (2) its heuristic function relies on an FF-style heuristic to rank which flaw to fix next; (3) if it cannot find a solution after a long time it switches to FF. Because it starts with a relaxed-plan of size equal to the final plan, LPG can outperform FF since it may require fewer fixes than FF, which starts from an empty plan.

The main difference between M and Mp is the SAT variable selection. While M uses a general SAT solver's algorithm, Mp sets goal orderings and prioritizes actions that achieve them. This ordering technique does not work well in UHP because there is little order between goals, especially since we allow any node to be visited first.[2]

Results on GC-inspired planning problems

We now turn to results on scheduling-type planning problems. Figure 3 shows the median running time of the FF planner for the planning problems inspired by 3-color graph coloring problems. Like Figure 1 for the UHP domain, Figure 3 (Top) confirms the solvable/unsolvable phase transition. The phase transition is sharp already at problem size $n = 18$, where n is the number of vertices in the graph (or

[2]We have preliminary results showing that this goal ordering technique is effective for Directed Hamiltonian Path problems which, being based on directed graphs, have more inherent order.

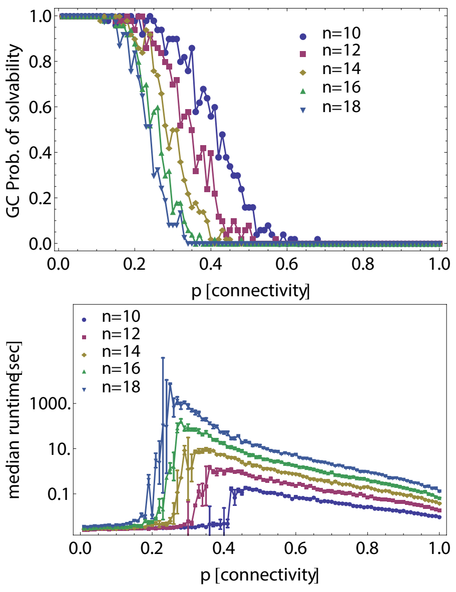


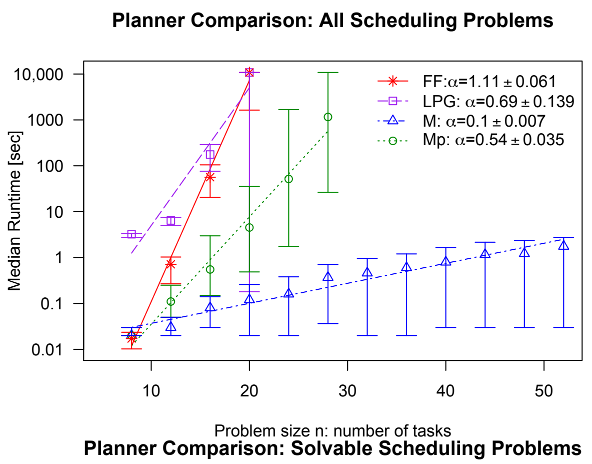
Figure 3: Scheduling-type planning problems. **Top:** Fraction of solvable instances as a function of the connectivity parameter p for different problem sizes. **Bottom:** Median runtime vs. p. The results shown are for the FF planner.

$3 \times n = 54$ grounded actions in the corresponding planning problem). Figure 3 (Bottom) shows that the typically hardest problems do indeed occur at the phase transition. The figure shows results for FF, but the other planners behave similarly.

To compare the planners, we created test sets of 1000 problems at each size (increment of 4) at the phase transition. A sharp phase transition threshold in the k-colorability of $G(n, p)$ graphs has been established for all $k \geq 3$ in terms of the parameter $c = m/n = p \times n$, the ratio of the number of edges to the number of vertices (Achlioptas and Friedgut 1999). The threshold scales as $c = k \log k$ in the leading term, but the precise location of this threshold is still an open question, even for $k = 3$ (Coja-Oghlan 2013).

Our runs were done with $c = 4.5$, a value intermediate to the best current lower bound (Achlioptas and Moore 2003) and upper bound (Dubois and Mandler 2002) for the phase transition. The median runtime for different planners on problems at this phase transition are shown in Figure 4 (Top), with error bars at the 35th and 65th percentile. We use a three-hour (10800 sec) cutoff. Once the median runtime reaches the cutoff, we no longer show results for that planner. For points with high median runtimes, the error bars are cut off at 10800 seconds. In contrast to the results for navigation-type problems, the M planner significantly outperforms other planners on these scheduling-type problems.

Figure 4 (Bottom) shows the runtime of different planners on the subset of solvable problems. As in the UHP domain, the only planner that changed its relative performance significantly when restricting to solvable instances is LPG. This is again due to its incomplete local search algorithm that performs badly on unsolvable instances.



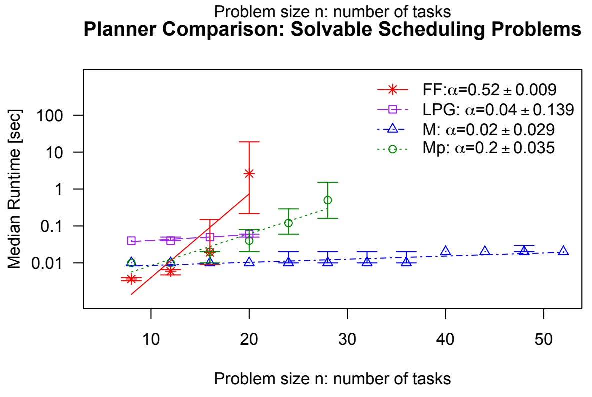


Figure 4: Median runtime vs. problem size at the phase transition for scheduling-type planning problems. (Top) All problems. (Bottom) Solvable problems.

Planners Comparison Analysis: The structure of graph coloring problems and navigation-based problems is markedly different. For example, multiple actions can be executed in parallel (e.g. two actions that color two unconnected nodes can be executed in parallel). As a result, the ranking of the planners is markedly different between the two cases. That actions can be executed in parallel favors compilation-based planners such as M and Mp which can find solutions at much lower planning horizons than FF which always returns a sequential plan, in this case of depth n. The Mp planner tries to build and utilize goal orderings, but this approach is poorly suited to this problem since there is no preferred order for coloring the nodes. Thus, Mp does not perform as well as the M planner. The median solving time of the LPG planners seems to suffer badly from the unsolvable instances (Figure 4). When faced with only solvable instances, its performance improves vastly (Figure 4 (Bottom)), especially over FF. One explanation is that LPG also uses FF's relaxed-plan heuristic but benefits from not having to sequentially search to a certain depth and enjoys fewer mutual exclusion constraints in this domain.

Solvable vs. Unsolvable:

When faced with a real-world planning problem, planners do not know whether it is solvable or not. Yet all existing benchmarks contain only guaranteed solvable instances. The easy conclusion would be that, among problems with similar characteristics, state-of-the-art planners have been honed to solve solvable problems better than unsolvable ones. Our results generally support this conclusion, but differ greatly between the two domains.

UHP: Figure 2 reveals that each planner takes roughly the same amount of time on solvable and unsolvable navigation problems; the median runtime of all planners improves by at most one order of magnitude when the unsolvable instances are removed. The uncertainty in expected runtime, however, decreases for all four planners. The change is most pronounced for the LPG planner which has the most predictable runtimes on the solvable instances and the least predicatable on the whole set. As we explained in the previous section, the incomplete local search algorithm LPG employs is badly equipped for unsolvable instance.

Graph Coloring: Figure 4 reveals that all four planners perform better on solvable instances of scheduling problems than on unsolvable ones. Median runtime reduced by an order of magnitude for the M, 3 orders of magnitude for Mp and FF, and 5 order of magnitude for LPG, moving LPG to 2nd place. The exponential slopes α also improved significantly, so the improvement is likely even more pronounced on larger problems. As for the UHP domain, the uncertainty in runtime reduced greatly for all planners, with LPG experiencing the most significant reduction. Figure 3 also illustrates FF's greater difficulty with unsolvable instances than solvable ones, even far from the phase transition.

Conclusions and Future Work

Our parametrized families of navigation-type and scheduling-type planning problems complement current benchmark sets obtained from real world applications. The analysis shows the sort of insights one can gain by examining planner performance on even small instances in these hard problem families. Our results suggest that one key reason for differences in planner performance is that some planners perform well on the scheduling aspects of planning problems but not so well on the navigation aspects, while others do the reverse. The improved planners of the future will need to perform well on both aspects.

Our families enable designers to tease apart reasons for their planner's performance, by giving them a sense for which aspects of planning problems their planners perform well on and which not. It would be exciting to see a future planner outperform the current best planners on both the scheduling- and navigation-type planning problems. As another example, our inclusion of unsolvable instances enables the evaluation of planners in the more realistic scenario in which it is unknown whether a plan exists. We hope this work spurs the development of more parametrized families of planning problems, from other NP-complete problems or through other means, that capture other aspects common to many planning problems.

Acknowledgements

Many thanks to Jüssi Rintanen for sharing his code for the generation of his hard family of problems, for making his code for the M and Mp planners available, and for generously answering our questions. Thanks also to Dimitris Achlioptas for sharing his knowledge about the state-of-the-art in phase transitions for graph coloring problems. We are grateful to Vadim Smelyanskiy and Sergey Knysh for useful discussions throughout the course of the work, to Chris Henze for his generous assistance with the NASA's Pleiades supercomputer, and to Bryan O'Gorman for helpful comments on drafts of this paper.

References

Achlioptas, D., and Friedgut, E. 1999. A sharp threshold for k-colorability. *Random Structures and Algorithms* 14(1):63–70.

Achlioptas, D., and Moore, C. 2003. Almost all graphs with average degree 4 are 3-colorable. *Journal of Computer and System Sciences* 67(2):441–471.

Bylander, T. 1996. A probabilistic analysis of propositional STRIPS planning. *Artificial Intelligence Journal* 81:241–271.

Cheeseman, P.; Kanefsky, B.; and Taylor, W. M. 1991. Where the really hard problems are. In *IJCAI*, volume 91, 331–337.

Chien, S.; Johnston, M.; Frank, J.; Giuliano, M.; Kavelaars, A.; Lenzen, C.; Policella, N.; and Verfailie, G. 2012. A generalized timeline representation, services, and interface for automating space mission operations. In *12th International Conference on Space Operations*.

Coja-Oghlan, A. 2013. Upper-bounding the k-colorability threshold by counting covers. arXiv:1305.0177.

Culberson, J.; Beacham, A.; and Papp, D. 1995. Hiding our colors. In *Proceedings of the CP95 Workshop on Studying and Solving Really Hard Problems*, 31–42.

Dubois, O., and Mandler, J. 2002. On the non-3-colourability of random graphs. arXiv:math/0209087.

Erdős, P., and Rényi, A. 1960. On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl* 5:17–61.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research* 20:239–290.

Helmert, M. 2003. Complexity results for standard benchmark domains in planning. *Artificial Intelligence Journal* 219–262.

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

Hoffmann, J. 2005. Where ignoring delete lists works: Local search topology in planning benchmarks. *Journal of Artificial Intelligence Research* 24:685–758.

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

Kautz, H. A., and Selman, B. 1999. Unifying sat-based and graph-based planning. In *Proceedings of IJCAI'1999*.

Komlós, J., and Szemerédi, E. 1983. Limit distribution for the existence of Hamiltonian cycles in a random graph. *Discrete Mathematics* 43(1):55–63.

Long, D., and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *J. Artif. Intell. Res.(JAIR)* 20:1–59.

Porco, A.; Machado, A.; and Bonet, B. 2011. Automatic polytime reductions of NP problems into a fragment of STRIPS. In *ICAPS*, 178 – 185.

Richter, S., and Westphal, M. 2010. The LAMA planner: guiding cost-based anytime planning with landmarks. *Journal of Artificial Research* 39:127–177.

Rintanen, J. 2004. Phase transitions in classical planning: an experimental study. In *Proceedings of the 14th International Conference on Automated Planning & Scheduling (ICAPS-2004)*, 101–110.

Rintanen, J. 2012a. Generation of hard solvable planning problems. In *Technical Report (TR-CS-12-03), Australian National University.*

Rintanen, J. 2012b. Planning as satisfiability: Heuristics. *Artificial Intelligence* 193:45–86.

Selman, B.; Mitchell, D. G.; and Levesque, H. J. 1996. Generating hard satisfiability problems. *Artificial intelligence* 81(1):17–29.

Slaney, J., and Thiebaux, S. 1998. On the hardness of decision and optimization problems. In *Proc. of the 13th European Conference on Artificial Intelligence*.