What Makes Planners Predictable?

Mark Roberts **Adele E. Howe**
Computer Science Dept.
Colorado State University
Fort Collins, Colorado 80523
{mroberts, howe}@cs.colostate.edu
http://www.cs.colostate.edu/meps

Brandon Wilson **Marie desJardins**
Dept. of Computer Science and Elec. Eng.
University of Maryland Baltimore County
1000 Hilltop Circle
Baltimore, MD 21250
{bwilson1,mariedj}@umbc.edu

Abstract

In recent work we showed that models constructed from planner performance data over a large suite of benchmark problems are surprisingly accurate; 91-99% accuracy for success and 3-496 seconds RMSE for runtime. In this paper, we examine the underlying causes of these accurate models. We deconstruct the learned models to assess how the features, the planners, the search space topology and the amount of training data facilitate predicting planner performance. We find that the models can be learned from relatively little training data (e.g., performance on 10% of the problems in some cases). Generally, having more features improves accuracy. However, the effect is often planner-dependent: in some cases, adding features degrades performance. We identify that the most prominent features in the models are domain features, though we find that the runtime models still have a need for better features. In the last part of the paper, we examine explanatory models to refine the planner dependencies and to identify linkages between problem structure and specific planners' performance.

Introduction

Performance is a function of the planner, its input (domain and problem description) and the platform on which it runs. Advances in planning support solution of old problems faster and new previously unsolved problems. Thus, performance is usually measured by number or percentage of problems solved, and by time to solution.

The nature of the performance function is not well understood, even for well established classical, STRIPS planning. Analyses of the International Planning Competitions (IPC) have identified relationships between the planning entries and the problems at the time (Long & Fox 2003; Hoffmann & Edelkamp 2005). By examining the local search topology, Hoffmann produced both theoretical and empirical analyses showing that many domains could be easily solved by heuristic search planners because either they had few local minima or the heuristic was well suited to them (Hoffmann 2004; 2005).

Today's planners are large and complex (e.g., FF is about 20,000 lines of C). IPC problem sets have been constructed to include increasingly difficult problems within a varied set of domains. Early on, the number of objects or goals was increased and found to be loosely correlated with difficulty (Bacchus 2001); more recently, Long and Fox found such variables to be poorly correlated with difficulty and highly dependent upon the planners (Long & Fox 2003).

To assess the state of the art in planning, an earlier study (Roberts & Howe 2007; forthcoming) collected performance data for STRIPS (and some ADL) planners on a large suite of problems. As part of that study, machine learning tools were used to learn models that could predict, given problem/domain features, the success or runtime for a planner on a particular problem. The learned models of success were fairly accurate (mean of 97% accuracy, which was equal to or better than the baseline of guessing failure, sometimes much better); the learned models for time were less accurate (RMSEs of 3.5–496 seconds).

The focus of this paper is determining what factors help to predict performance and why performance is as predictable as it is. Given the complexity of the planners and large suite of problems, one might expect it to be difficult to predict performance. Models of the performance function can be used to allocate time among the planners (such as in the portfolios of BUS (Howe *et al.* 1999) and HAP (Vrakas *et al.* 2005)), and may lead to a better understanding of which components of planners work well, and which could be improved. Consequently, we ignore the impact of different platforms and study other factors: how many and what types of problems, what types of models, what types of features and what search topology. Our results suggest that domain is paramount in predicting success (even with few examples relative to the dataset), but not enough for predicting time. The models appear to generalize across domains, although the influence of specific factors on predictability is idiosyncratic to particular planners. Finally, we examine some explanatory models to try to uncover some underlying causality.

Learning the Models

We obtained the performance data from the earlier study (Roberts & Howe forthcoming). The performance data set was derived by running 28 publicly available planners on 4726 problems from 385 domains, which are represented in PDDL, using the same hardware/software platform. For each planner/problem combination, performance is measured as: whether the planner succeeded, failed or timed out

Planner	Success	TO	Fail	Ratio	Prior	μ time
HSP-2.0r-h1plus	71	88	1269	0.05	0.95	179.4
OPTOP	86	2	1340	0.06	0.94	63.9
BlkBox-4.2	87	3	1338	0.06	0.94	21.6
R-1.1	107	303	1018	0.07	0.93	429.7
MIPS-3	110	252	1066	0.08	0.92	366.7
LPG-1.1	166	127	1135	0.12	0.88	211.7
Satplan04	194	64	1170	0.14	0.86	137.2
HSP-2.0	197	283	948	0.14	0.86	418.7
IPP-4.0	202	247	979	0.14	0.86	412.5
SimPlan-2.0	203	2	1223	0.14	0.86	40.7
IPP-4.1	210	229	989	0.15	0.85	406.1
LPG-1.2	217	109	1102	0.15	0.85	199.2
Metric-FF	566	10	852	0.40	0.60	56.4
LPG-TD	596	107	725	0.42	0.58	263.6
FF-2.3	602	76	750	0.42	0.58	139.1
FastDown	694	28	706	0.49	0.51	133.6
SGPlan-06	708	39	681	0.50	0.50	175.7

Table 1: The 17 planners, sorted by success rate (Ratio), including counts of outcomes and mean time to termination.

(using a 30-minute cut-off) and how much time was required to terminate. For the purpose of this paper, we restrict our analysis to a subset of the problems (those that are challenging) and a subset of the planners (those that solved enough challenging problems to support analysis).

Problems and planners The full problem collection was formed from Hoffman's problems (Hoffmann 2004), the UCPOP Strict benchmark, IPC sets (IPC1, IPC2, IPC3 Easy Typed, IPC4 Strict Typed and IPC5 Propositional) and 37 other problems from two other available domains (Sodor and Stek). The majority of these problems appeared to be easy—that is, solvable by most planners in under one second. To make the prediction task more interesting, we define *challenging* problems to be those that can be solved by only one, two or three planners or that showed a median time for solution of greater than one second. The reduced set contains 1428 problems from 47 domains, with most of the problems coming from the more recent problem sets[1].

The full planner set included some old planners that were not able to solve more recent problems. Thus, for them, the best predictive model of success is simply to predict failure *always*. We restricted our set to the 17 planners with a success rate of at least five percent on the challenging problems.

The features Each problem instance is defined by 19 features that can be automatically extracted from PDDL problem and domain definitions. We started with features from previous work (Howe *et al.* 1999; Hoffmann 2001). We added others that matched our intuition and removed those that were only computationally tractable for small problems. Table 2 lists the features in two categories: domain-specific at the top and instance-specific at the bottom.

[1]We welcome additions to this problem set.

Metrics	Description
num	# of operators
num	# of predicates
min,mu,max	arity for predicates
min,mu,max	predicates in precondition
min,mu,max	predicates in effects
min,mu,max	negations in effects
num,ratio	actions over negative effects
num	# of goals
num	# of objects
num	# of inits

Table 2: The feature set: first column is the type of measure being collected, the last column briefly describes the feature, domain features at top, problem instance features at bottom.

Learned predictive models For each planner, we constructed a discrete classification model for success and a quantitative model for time. We tried 32 different models from WEKA (Witten & Frank 2005), a commonly used toolkit of machine learning algorithms. Unless we are testing a specific hypothesis that requires a particular model, we report the results for the best of the models found.

The baseline models are built from the 19 features in Table 2. The baseline success models define two classes: {solve/fail-to-solve} (timed-out (TO) runs are in fail-to-solve). We present average accuracy over ten-fold cross-validation in the column labeled *SF* (for Success/Fail) of Table 3. The baseline runtime models predict runtime as a continuous value; the results for these models are shown in the column labeled *SFC* (for Success/Fail/Censored) of Table 4.

Influential Factors in the Predictive Models

Our goal is to determine some of the factors that influence the predictability of planners. We accomplish this indirectly by augmenting or ablating the baseline models and considering the impact of the change on model accuracy. We define model accuracy as percentage correct for the success models and as RMSE (root mean squared error) for the time models.

The factors are: success as a binary or ternary classification, different types of features and number of examples that are used for training. We test the contribution of each factor using a two-tailed, paired-sample t-test to determine if a change in accuracy is detected. A statistically significant difference in accuracy ($p < 0.05$) identifies further lines of inquiry for what is influencing model performance.

We report the results in Tables 3 and 4. Accuracy results are reported relative to a naive model that, for success, uses the prior probability of failure across the entire data set ("Prior"), and for time, the mean time to solution across the entire data set ("mean.time"). Because some of the planners have failure rates of $> 92\%$ on these problems, it could be misleading to report a model accuracy of 90%, for example, which might on its own be viewed as good. So with a prior of 95%, a 90 would be listed in the table as -5. Because we are maximizing percentage correct, positive numbers are better for success; negative numbers are better for RMSE (time). All t-tests are performed with respect to the most

complete classifier overall, which is SF for success models and SFC for time models (see below for their descriptions).

Should time-outs be handled separately?

Including timed-out (TO) runs as failures may bias the learners away from good models because a planner may have solved the problem, given more time. A timed-out run is considered to be *right-censored*, a distinction inspired by survival analysis (Elandt-Johnson & Johnson 1980). To answer the question of whether time-outs matter, we model success as a ternary variable.

Column SFC of Table 3 shows that accuracy drops a modest amount overall when predicting success as a ternary response. A paired-sample t-test indicates that the difference is significant ($t = -4.90, p \approx 0.000$). It seems likely that TO is not that distinguishable from failed cases.

Which features contribute to accuracy?

We hypothesize that the function relating planner to performance is fairly complex and so more features are needed for better predictions. In this section, we examine the contribution of different types of features: outcome, PDDL requirements, problem instance, domain, and causal graph.

"Success" in the time models The baseline runtime model ("SFC" column) includes only features extracted from the problem and domain description. We observe high variance in the runtimes with distributions that appear log-normal with long tails, and that planners tend to fail quickly (e.g., ran out of memory), except for time-outs.

Consequently, for a more comprehensive model, we tried adding binary success (whether a run succeeded or failed) as a feature (the SF column of Table 4). We find that including success lowers mean RMSE by 45 seconds on average compared to the No Success (NS) model. Using the ternary success feature (in Table 4, column SFC) significantly improves the average RMSE of the models by about 32 seconds over SF ($t = 3.11, p \approx 0.007$). Because SFC is more complete and accurate than the SF model, we use it as the comparator for subsequent t-tests.

Note that we are using the *actual* (rather than predicted) values for success. This choice was guided by our questions about the value of including success as a feature and whether time-outs matter in light of that. As a post-hoc analysis, this allows us to address the questions and (in the future) justify layered models that would rely on a predicted value for success. We felt that using predicted success added a confounding layer to understanding the impact of using success as a feature.

Language requirements Some planners handle only some of the PDDL/ADL requirements. Thus, one might expect particular language requirements to be critical to performance for some planners. On the other hand, some domain descriptions may or may not correctly report their requirements, which may offset any potential advantage.

We examined 13 language features that follow the requirements list of PDDL (ADL, fluents, typing, etc.). To test their contribution, we add them into the feature set (that now totals 32 features) and train the models again (see the LF columns of Tables 3 and 4). The overall effect was slightly (but not significantly) beneficial for runtime ($t = 1.41, p \approx 0.178$) and significantly worse for success ($t = -6.09, p \approx 0.000$). The success results suggest where there may be errors in the stated language requirements for a domain. For runtime, it suggests that either the benefits and problems cancel out or that other features subsume these; we view the second as more likely because we would have expected the STRIPS planners to be more sensitive to these features.

Problem instance features The feature set is heavily skewed toward characterizing the domain: 16 of the features are extracted from the domain file, while only three are from the problem instance file. To determine the contribution of domain and problem features, we constructed models using only each feature subset separately and compared them to SF (success) or SFC (time). Tables 3 and 4 list the results under the columns labeled DOF (domain-only) and IOF (instance-only) features.

For the success models, both feature subsets produce significantly less accurate models. The domain features suffer slightly less of a loss (3.17 versus 4.77 average accuracy loss). So it seems that they are are more critical to success, possibly because there are fewer problem features. Also, prior research shows how domains can influence the structure of the search space (Hoffmann 2005). However, all of the IOF models improve over the priors, which refutes the hypothesis that the predictions are simply based on learning to distinguish the domains.

For the runtime models, both subsets again degrade performance, but neither is significant. As with success, the domain features account for most of the model accuracy; the DOF column shows the average difference in RMSE to be only 17.32 seconds worse. However, again, the SFC model appears to be using more than just knowledge of the domain to reduce the error.

Should fast runs be handled separately? Even though we used a subset of the more challenging problems, about 62% of the planner runs still finish in under a second. We analyze whether models trained on the full set differed from models trained on just the instances taking over one second. The SF.OO and SFC.OO columns of Tables 3 and 4 list the results for models trained on the full data and tested on the Over One (OO) data, while the OO.OO models are trained and tested only on the Over One data. In both cases the success models improved by an average of about 1% accuracy, although only the OO.OO model was significantly better. For the time models, both cases were significantly better, but the average difference in RMSE was higher for the OO.OO model. These results suggest that there appears to be a qualitative difference between the runs that are over and under one second.

	Learner	Prior	SF	SFC	LF	DOF	IOF	SF.OO	OO.OO	SF.CG	DICG	I.CG
BlkBox-4.2	rules.PART	94	4.2	3.7	0.0	3.5	0.0	4.9	3.4	3.6	3.5	3.2
FastDown	trees.RandomForest	51	43.1	42.7	32.4	37.7	32.4	48.1	48.4	43.0	42.7	40.6
FF-2.3	trees.J48	58	37.3	35.9	28.3	33.5	28.3	29.4	36.0	35.4	35.1	33.8
HSP-2.0r-h1plus	lazy.IB1	95	3.2	-2.1	0.2	0.0	0.2	5.0	5.0	3.1	2.9	1.7
HSP-2.0	trees.LMT	86	8.5	4.1	3.2	4.8	3.2	7.2	7.0	8.8	6.8	7.0
IPP-4.0	trees.RandomForest	86	11.5	4.8	10.7	10.6	10.7	13.5	13.5	11.8	11.7	11.7
IPP-4.1	trees.RandomForest	85	12.7	6.4	12.1	11.4	12.1	14.6	14.7	12.2	12.6	12.2
LPG-1.1	trees.RandomForest	88	8.4	6.0	4.9	4.9	4.9	11.5	11.2	7.6	8.0	7.7
LPG-1.2	lazy.KStar	85	11.2	9.7	5.9	5.7	5.9	9.3	11.3	10.9	9.8	8.3
LPG-TD	lazy.KStar	58	36.0	33.3	27.3	32.5	27.3	39.9	40.4	36.0	34.5	32.4
Metric-FF	lazy.KStar	60	34.1	33.5	28.7	29.6	28.7	35.8	37.4	33.1	33.5	33.6
MIPS-3	trees.RandomForest	92	4.6	1.8	4.1	1.0	4.1	7.0	7.2	3.1	3.6	3.8
OPTOP	lazy.IB1	94	3.1	2.6	1.4	0.9	1.4	5.5	5.5	2.9	3.6	3.8
R-1.1	lazy.IB1	93	6.4	2.2	4.8	5.6	4.8	7.0	7.0	6.7	6.7	4.5
Satplan04	lazy.KStar	86	9.4	8.5	1.6	6.4	1.6	12.3	12.6	9.5	10.0	11.1
SGPlan-06	trees.LMT	50	41.5	39.8	33.3	38.7	33.3	45.0	39.2	38.4	40.3	40.4
SimPlan-2.0	trees.LMT	86	6.6	6.2	2.0	1.4	2.0	6.0	4.2	7.4	7.6	6.2
t.score			—	-4.90	-6.09	-8.32	-6.09	1.62	2.31	—	-0.19	-1.80
pVal			—	0.000	0.000	0.000	0.000	0.125	0.035	—	0.854	0.091
t.meanDiff			—	-2.53	-4.77	-3.17	-4.77	1.17	1.31	—	-0.04	-0.70

Table 3: Accuracy results for the success models (out of 100) as compared to the prior. All comparisons are made against SF. The final three columns present results for the subset of the data for which we could calculate the causal graph features (described in the text). The bottom three rows indicate the t-test results in the form of the test statistic (t.score) the significance (pVal) and the mean difference (t.meanDiff) from the baseline. Baselines for the tests are shown with '—' and tests to the right are against those baselines to the left.

	Learner	μ time	NS	SF	SFC	LF	DOF	IOF	SF.OO	OO.OO	SFC.CG	DICG	I.CG
BlkBox-4.2	lazy.KStar	21.64	97.3	89.5	86.0	86.1	79.0	85.9	8.2	5.2	80.5	83.3	89.9
FastDown	lazy.KStar	133.62	79.1	-7.1	-6.8	-7.4	13.9	-9.3	-84.9	-84.9	14.4	96.7	93.7
FF-2.3	lazy.KStar	139.08	135.3	85.8	85.1	85.2	81.2	89.4	-1.5	-27.9	105.5	130.7	147.3
HSP-2.0r-h1plus	lazy.KStar	179.35	93.6	84.6	-1.7	-5.1	2.3	36.9	-25.2	-32.2	27.9	86.8	109.0
HSP-2.0	lazy.KStar	418.74	-59.3	-154.0	-189.2	-189.4	-176.3	-175.1	-265.5	-281.7	-180.1	-35.2	-6.9
IPP-4.0	lazy.KStar	412.52	-110.4	-133.1	-194.0	-191.9	-98.6	-157.1	-247.1	-260.5	-197.6	-97.0	-103.6
IPP-4.1	lazy.KStar	406.07	-110.6	-130.3	-185.8	-182.2	-81.0	-171.0	-265.2	-276.7	-189.0	-91.2	-118.3
LPG-1.1	lazy.KStar	211.65	9.2	-24.2	-37.3	-36.5	12.2	-16.5	-74.1	-109.7	-35.0	-26.3	47.6
LPG-1.2	lazy.KStar	199.15	30.8	-24.8	-34.9	-34.5	3.2	1.6	-54.6	-72.9	-30.9	53.2	111.8
LPG-TD	lazy.KStar	263.59	44.6	-28.9	-26.0	-25.6	-37.2	15.0	-200.4	-202.5	-11.2	84.8	101.4
Metric-FF	lazy.KStar	56.40	118.6	89.1	75.6	75.8	83.8	80.3	34.7	19.3	113.2	126.3	129.8
MIPS-3	lazy.KStar	366.67	-82.3	-114.1	-147.6	-143.6	-157.0	-143.7	-222.4	-232.5	-131.2	24.1	50.5
OPTOP	lazy.KStar	63.91	-8.1	-9.2	-9.2	-9.2	-7.6	32.8	-51.5	-51.5	-8.0	-3.0	4.9
R-1.1	lazy.KStar	429.67	-252.9	-275.7	-300.4	-300.1	-301.9	-217.7	-373.7	-375.1	-292.2	-240.9	-155.7
Satplan04	lazy.KStar	137.15	151.8	129.6	107.5	109.5	135.0	43.7	-69.6	-75.1	53.4	97.1	76.5
SGPlan-06	lazy.KStar	175.70	650.3	454.8	291.6	291.8	225.0	593.9	28.3	1.5	273.5	727.4	855.6
SimPlan-2.0	lazy.KStar	40.69	72.3	61.7	42.9	42.9	75.1	67.5	19.0	14.3	55.9	74.9	92.3
t.score.row			4.00	3.11	—	1.41	1.78	1.94	-5.23	-5.82	—	3.29	3.42
pVal			0.001	0.007	—	0.178	0.095	0.070	0.000	0.000	—	0.005	0.004
t.meanDiff			76.68	31.63	—	0.57	17.37	35.34	-82.43	-94.05	—	84.87	110.40

Table 4: Accuracy results for the runtime models (as RMSE in seconds) as compared to the mean runtime across all planners. All comparisons are made against SFC. The final three columns present results for the subset of the data for which we could calculate the causal graph features (described in the text). The bottom three rows indicate the t-test results in the form of the test statistic (t.score) the significance (pVal) and the mean difference from the baseline (t.meanDiff). Baselines for the tests are shown with '—' and tests to the right are against those baselines to the left.

Causal graph features The analysis of problem instance features above indicates that problem features do contribute to predictions, even when we only have three problem features. We consider another class of problem features based on computing metrics from the causal graph (CG). We obtained the code to compute the CG from the FastDownward planner (Helmert 2006a). The features we extracted characterize the number of variables, operators, axioms, and vertices in the causal graph after its translation from PDDL to SAS+. We also include the ratio of edges to a fully connected graph, the average degree, and the ratio of back edges that persist after a (pseudo) topological sort of the graph. Finally, we include a boolean variable indicating whether the graph is acyclic and can be solved in polynomial time.

We use the subset of 1007 problems for which the causal graph can be computed with 2 GB of memory in less than 30 minutes. We assess the impact of these features in the last three columns of Tables 3 and 4, where the baseline results for the 1007 problems are shown in columns SF.CG and SFC.CG, respectively.

The results for the success models are insignificant, which suggests that the CG features neither add nor detract from the success models. Although the loss of accuracy is less for the I.CG models than the IOF models. But for runtime, using all Domain-Instance-CG features (the DICG column) is significantly worse than the baseline by an average of about 85 seconds. When only the instance and CG features (I.CG) are used, there is an even greater (and statistically significant) drop in accuracy of about 110 seconds, which suggests the bulk of learning is coming from the domain features. These results suggest that the causal graph features either do not help or distract the models, an observation that runs very counter to our expectations given the strength of FastDownward in recent competitions. We intend to examine this further in future work.

How many instances are required?

The next factor is the influence of the amount of data, i.e., number of examples. To test this question we randomly select a hold-out testing set, S, of 20 percent of the instances. The remaining 80 percent are randomly partitioned into training data of 10 (approximately) equal-sized bins ($b_1..b_{10}$). From these bins, we create training sets, $T_j, j = (1..10)$ such that T_j includes all bins from 1 to j ($T_j = \bigcap_{i=1}^{j} b_i$). We train models (SF for success and SFC for time) on each T_j and test on the set S. This process is repeated ten times to cross-validate the results, and the final statistics are gathered from these ten runs.

Our results showed that model accuracy does decrease with less training data, but that the decrease in accuracy is not as pronounced as one might have expected. For example, for a representative planner, FF-2.3, the failure rate for FF-2.3 is 46 percent, and mean time is 141.48 seconds. When only 10% of the instances (approximately 100 problems) are used for training, the success accuracy for runtime and success models is still well above the prior, even for the worst of the ten runs.

Accuracy improves with more training instances. However, accuracy improvement levels off after about 80% of the training data. Given the number of combinations of feature values and domains represented, it is somewhat surprising that performance can be predicted well using only 130 problems. Again, this may be due to the dominance of domain features as predictors; only one instance per domain is needed to roughly predict for it. Indeed, at 10% of the training data about 30 of the 47 domains have at least one instance represented.

Analysis of Domain in Prediction

The interaction of the domain and the planner is the dominant factor in predicting performance. Every planning competition and many studies of planner performance are predicated on this. In this section, we analyze more fully how the domain properties affect the accuracy results.

Do the models simply learn to predict "domain"?

As shown in the last section, the problem features do contribute to the success of predictions, but their influence is not strong. We re-visit the issue of whether the models are essentially learning the domain by first showing that it is possible and second showing that it is valuable to do so.

To show that it is possible, we use the feature set to explicitly learn the domain. We follow the same procedure as with performance models except that the classification was the 47 domains. We find that the best model can classify the domain with better than 99 percent accuracy. Several of the domain features nearly uniquely define the domains; the continuous values in the mean features (e.g., mu predicates in precondition) tend to separate by domain.

To show that it is valuable, we construct models using only the domain as a feature. Table 5 shows the accuracy results on these models. The Domain Name Only (DNO) models for success are not as accurate as the SF models (-5.5% drop in average accuracy); Δ SF column lists the difference in accuracy between the DNO and SF (negative means DNO is worse). The DNO time models are much worse (positive means DNO is worse) by about 60 seconds on average than the SFC baseline. Thus, determining domain essentially determines success, but does not determine time.

Do the models generalize across domains?

While most domains separate on key domain feature values, a few share these values, e.g., the propositional domains all have arity 0 for the predicates. Displaying the performance data for these domains suggests that performance may be similar as well – these domains may place similar demands on the planners and support some generalization.

One common relationship between the domains is seen in Depots (27 problems in the Challenge3 set), Logistics (15 problems), and Blocks (69 problems) because Depots is a combination of Logistics and Blocks. To see whether performance in one generalizes to another, we learn models in both directions. First, we train models on the Depots problems then test them on Blocks/Logistics. Second, we train

	Success			Time		
Planner	Prior	DNO	Δ SF	μ Time	DNO	Δ SFC
BlkBox-4.2	94	0.1	-4.3	21.6	-102.0	15.9
FastDown	51	-33.6	-9.5	133.6	-106.0	112.8
FF-2.3	58	-30.5	-6.8	139.1	-85.8	0.7
HSP-2.0r-h1plus	95	1.5	-4.7	179.3	-59.4	61.1
HSP-2.0	86	-3.5	-5.0	418.7	137.1	52.0
IPP-4.0	86	-10.6	-0.9	412.5	95.6	98.4
IPP-4.1	85	-11.4	-1.3	406.1	79.5	106.3
LPG-1.1	88	-5.0	-3.4	211.7	-23.4	60.7
LPG-1.2	85	0.3	-11.5	199.2	-40.2	75.2
LPG-TD	58	-27.9	-8.1	263.6	-33.6	59.6
Metric-FF	60	-25.9	-8.2	56.4	-107.9	32.3
MIPS-3	92	-0.2	-4.4	366.7	122.4	25.2
OPTOP	94	0.8	-3.9	63.9	-57.7	66.9
R-1.1	93	-4.0	-2.4	429.7	216.8	83.6
Satplan04	86	-0.4	-9.0	137.2	-73.2	-34.2
SGPlan-06	50	-36.7	-4.8	175.7	-451.2	159.6
SimPlan-2.0	86	-1.7	-4.9	40.7	-102.0	59.1

Table 5: Accuracy of success and time models generated from just Domain Name Only (DNO) relative to the priors for success (Table 3) and runtime (Table 4).

models on Blocks/Logistics problems and test them on Depots.

For the success models trained on Depots, 10 (of 16) models had worse accuracy than the prior for Blocks and 6 were worse for Logistics; the gap in how bad ranged from 0 to 80 percent. But when trained on the combined Blocks/Logistics only 5 models had worse accuracy than the prior; the gap in how bad ranged from 4% to 27%. For the runtime models trained on Depots and tested on Blocks, 8 had worse RMSE (from 3–1840 seconds); when tested on Logistics, 11 were worse (from 7–1400 seconds). When trained on Blocks/Logistics and tested on Depot, 9 models were worse (from 17–460 seconds).

Even with the mixed results there seems to be a trend toward generalization across domains. The models trained on Blocks/Logistics perform better with lower variance than the other way around. But models trained on the more general Depots problems seem to have trouble with specific subproblems; this is probably due to the low number of problems for Logistics and Depots.

Is search topology a factor of model accuracy?

The most extensive topological analysis for classical planning is that of Hoffmann. To assess whether the taxonomy is a factor in the models, we group the problems by Hoffmann's taxonomy (Hoffmann 2005) and analyze how success model accuracy changes for particular planners.

We have summarized Hoffmann's taxonomy in Table 6. Domains either have local minima (MT) or do not (ML), and some domains that lack minima also have benches with a median exit distance less than a constant (MB). Along the dead-end axis, the topology divides domains according to the presence of dead-ends. If dead-ends do not exist, the transition graph is either undirected (HC) or directed but harmless (HH). When dead-ends exist, they are heuristically recognized (HR) or heuristically unrecognized (HU). Note that the ordering of the taxonomy categories listed in the caption implies that problems in the MT:HU pairing are among the most challenging while those in the lowest pairing MB:HC are among the most simple.

	HC	HH	HR	HU
MT	112	197	27	388
ML	1	4	0	x
MB	15	110	240	x

Table 6: A summary table of the number of domains from Challenge3 in each category of the taxonomy. The 'x' indicates an impossible category for the taxonomy.

Training	Testing Pair				
Pair	MBHH	MBHR	MTHC	MTHH	MTHU
MBHH	98.2	4.6	59.8	34.0	62.6
MBHR	16.4	100.0	25.9	52.8	69.8
MTHC	27.3	24.6	71.4	38.6	31.4
MTHH	34.5	28.8	25.9	86.8	51.8
MTHU	39.1	87.1	27.7	59.9	96.4

Training	Testing Pair				
Pair	MBHH	MBHR	MTHC	MTHH	MTHU
MBHH	99.1	67.1	89.3	39.6	72.4
MBHR	81.8	97.5	99.1	53.8	42.0
MTHC	55.5	20.4	99.1	70.1	17.0
MTHH	24.5	73.8	8.0	82.7	78.4
MTHU	40.0	65.0	7.1	66.0	98.2

Table 7: The success rates for training and testing on each combination of the taxonomy categories for which there were at least 100 problems. The top sub-table shows FF and the bottom sub-table shows LPG-TD.

We learned models of hType and mType based on the features from Table 2. Table 7 shows the results for FF (top) and LPG-TD (bottom). FF exhibits a trend that within a grouping (along the diagonal) the accuracy is high, but across groupings (the off-diagonal entries, among others), the accuracy drops considerably. This trend is similar for MetricFF and SGPlan (not shown), but not as strong for LPG-TD (bottom sub-table), which uses a heuristic distinct from the FF family. These results suggest that, in part, domain is still a strong indicator of performance, but that the taxonomy is also factor into the accuracy of the success models.

Explaining Model Performance

In this section, we develop explanatory models of planner performance, using a variety of machine learning techniques to analyze the influence of problem and domain features on performance. We summarize some key findings of three such techniques: single best feature (OneR), Cluster-Wise Linear Regression (CWLR) and feature selection. We do not discuss model accuracy here, but it is worth noting that in some cases, these models (especially CWLR) outperform the predictive models we have already discussed.

We focus the analysis on the four planners with the high-

est rates of success: FF, Metric-FF, SGPlan-06, and LPG-TD. For most analyses, to avoid the heavy bias of very short and very long (timed-out) runs, we also restricted attention to runs that completed between 60 and 1700 seconds.

Single best feature

OneR selects a single best feature for classification (Holte 1993). For six of 17 the planners, OneR provided the best model for predicting success. The feature selected by OneR for IPP-4.1 was num.invariant.pred; for LPG-1.1 and LPG-1.2, mu.post.oper; for MIPS-3, mu.prec.oper; for OPTOP, num.inits; and for SystemR, mu.neg.post.oper. These planners all had success rates below 20%, and for all but one (OPTOP), the selected feature was a domain feature, rather than a problem feature.

For the four planners with the highest rates of success, OneR selected mu.prec.oper as the predictor for success. The top graph in Figure 1 shows how the successes for FF-2.3 are strongly grouped by this feature; results for SGPlan, IPP-4.1, LPG-1.1, LPG-1.2, MIPS-3 and R-1.1 also looked similar to that of FF-2.3.

The mu.prec.oper feature does not yield a strong grouping for LPG-TD (bottom of Figure 1) or for MetricFF. As discussed in the last section, the (vertical) groupings are usually from the same domain. However, when we include runs outside the range of 60–1700 seconds, the groupings contain mixed groups, especially at the lowest value of the feature (which is 0 for many of the grounded/propositional versions from the IPC4 and IPC5 benchmarks).

Cluster-wise linear regression models

Linear regression uses a simple statistical model of quantitative feature interactions to fit a hyperplane through the set of data points in multi-dimensional space. The Cluster-wise Linear Regression Model (CWLR) combines K-means clustering with regression analysis to create clusters of regression functions that span the feature space.

CWLR, developed by Späth (Späth 1979), identifies multiple overlapping regression functions in a given data set. To train a CWLR model, the instances are initially partitioned into k random, disjoint clusters. The algorithm then iteratively generates a linear regression equation for each cluster and reassigns instances to the "best" cluster (i.e., the one whose regression equation predicts the target value with the smallest error). The iteration ceases after an iteration where none of the points are reassigned or a maximum of 50 iterations are performed. We select the k with the smallest RMSE. The final set of clusters are used to train a classifier for predicting cluster membership when new instances are encountered during testing.

Given the behavior of the success models, one might expect that the clusters would group around domain-like centroids. However, this was not the case. In fact, the number of clusters is much smaller than the number of domains: 3 for FF, 5 for LPG-TD, and 4 for MetricFF and SGPlan. The overall RMSE remained competitive with (sometimes better than) other predictive models.

To understand if domain was driving the cluster formation, we ran CWLR on the entire feature set (domain, plus instance, plus CG) then plotted the instances by the first two principal components. We then labeled the points according to their cluster classification but colored them by domain. The plots revealed strong linear associations between the principal components that mostly lined up with the clusters. However, the plots also showed clearly that the clusters are not centered strongly around domain; a single domain frequently shows up in multiple clusters. We believe this indicates that the runtime models are likely leveraging inter-domain knowledge that the success models are not using.

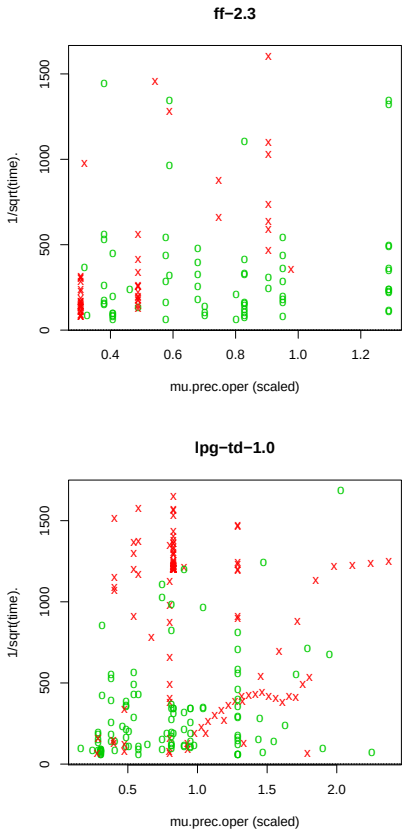

Figure 1: The transformed runtime of planner instances for FF-2.3 (top) and LPG-TD (bottom) as they relate to a feature that sometimes groups domains, mu.prec.oper, which has been scaled to a t-value. 'o' indicates a successful run, while 'x' indicates a failed run.

Further analysis of the regression models associated with the clusters shows that the clusters for some planners are far more accurate than the overall model accuracy indicates. This happens because the classifier is not able to reconstruct the concept that led to the original clustering. Accurate cluster models show that some planners have distinct behavior in different areas of the feature space. Therefore a single, accurate model of runtime actually requires several underlying models to predict performance in these different areas. Even though the CWLR model itself does not always make accurate predictions, the accuracy of the individual cluster

models serves to explain why the runtime for some planners is much more difficult to predict than success.

Feature selection

Feature selection focuses attention on the strongest or most individually salient features. We applied all-subsets selection using Mallow's C_p criterion, which favors smaller models (Kutner *et al*. 2005). The selection algorithm performs branch-and-bound search over all of the features (using the "leaps" package in R). The full model used the original 19 domain and instance features plus success as an indicator variable; note that the feature count (21) includes the intercept term. The response was log-transformed (we added a small constant to zero time entries). Most of the final models used almost all (17-19) of the features. Only one planner, R-1.1, used 13 features. The most commonly omitted feature was num.operators (used in only four models). Four features were always used: mu.params.pred, mu.post.oper, ratio.neg.post.oper, and success.

Discussion

Nearly all the analyses point to the importance of the domain in success predictability – a planner either solves the problems in a particular domain or it fails on them. Because we have a relatively small set of domains, each has a unique signature, as a combination of the domain features.

Planner time is harder to predict because time can vary significantly across problem instances. Times do somewhat separate by success. Both domain and instance features enhance predictability, but in a planner-dependent way.

Models for some of the planners show generalization across domain. For example, some planners do not need the full signature for the domain. We also find evidence of generalization across specific domains that are designed to be similar.

Hoffmann's topology and the explanatory models offer some guidance of how the domain drives performance. Such analyses must link the planner to the domain, as is done in connecting the h^+ heuristic to Hoffmann's problem taxonomy. The CWLR clusters suggest that the runtime models are using some underlying inter-domain knowledge.

Understanding the factors for predicting performance within and across domains remains a challenge – especially for the runtime models. Two issues must be explored further. First, the results of the explanatory models are intriguing, but clearly incomplete. We will be further analyzing what the clusters may reveal, considering other related analyses (e.g., Helmert's complexity analysis of the domains (Helmert 2006b)) and deriving additional features designed for specific planning algorithms/heuristics. However, this analysis is limited by the relatively small domain set, the lack of overlap in some of the key domain features and the difficulty of finding problems that can be solved in more than a second. Therefore, the second issue is that of the problem set. Given the availability of problem generators, we will first construct a wider set of instances that scale by parameters other than the traditional ones for specific domains, and will also work to collect and construct new domains to fill the gaps.

Acknowledgments We wish to thank the anonymous reviewers for insightful and thought provoking discussion of this work as well as for suggestions for its presentation. Adele Howe and Mark Roberts were partially supported by AFOSR grant FA9550-07-1-0403.

References

Bacchus, F. 2001. Aips'00 planning competition. *AI Magazine* 3(22):47–56.

Elandt-Johnson, R. C., and Johnson, N. L. 1980. *Survival Models and Data Analysis*. New York: Wiley and Sons.

Helmert, M. 2006a. The fast downward planning system. *Journal of Artifical Intelligence Research* 26:191–246.

Helmert, M. 2006b. New complexity results for classical planning benchmarks. In *Proc. of the ICAPS-06. AAAI Press, Menlo Park, California. Cumbria, UK. June 6-10.*, 52–61.

Hoffmann, J., and Edelkamp, S. 2005. The deterministic part of IPC-4: An overview. *Journal of Artificial Intelligence Research* 24:519–579.

Hoffmann, J. 2001. Local search topology in planning benchmarks: An empirical analysis. In *Proceedings of IJCAI'01.*, 453–458. Seattle, Washington, USA: Kaufmann.

Hoffmann, J. 2004. *Utilizing Problem Structure in Planning: A local search approach*. Berlin, New York: Springer-Verlag.

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

Holte, R. C. 1993. Very simple classification rules perform well on most commonly used datasets. *Machine Learning* 3:63–91.

Howe, A.; Dahlman, E.; Hansen, C.; Scheetz, M.; and von-Mayrhauser, A. 1999. Exploiting competitive planner performance. In *Proc. 5th European ECP-99. Springer, LNCS, 1809. Durham, UK, September 8-10.*

Kutner, M.; Nachtsheim, C.; Neter, J.; and Li, W. 2005. *Applied Linear Statistical Models*. McGraw Hill.

Long, D., and Fox, M. 2003. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research* 20:1–59.

Roberts, M., and Howe, A. 2007. Learned models of performance for many planners. In *Working Notes of ICAPS2007 Workshop on AI Planning and Learning*.

Roberts, M., and Howe, A. forthcoming. Learning from planner performance. *Artificial Intelligence*.

Späth, H. 1979. Algorithm 39 Clusterwise linear regression. *Computing* 22(4):367–373.

Vrakas, D.; Tsoumakas, G.; Bassiliades, N.; and Vlahavas, I. 2005. *Intelligent Techniques for Planning*. Idea Group. chapter Machine Learning for Adaptive Planning, 90–120.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical machine learning tools and techniques*. San Francisco: Morgan Kaufmann, 2nd edition.