Learned Models of Performance for Many Planners [*]

Mark Roberts, Adele Howe and **Landon Flom**
Computer Science Dept., Colorado State University
Fort Collins, Colorado 80523 U.S.A.
mroberts,howe,flom@cs.colostate.edu

Abstract

We describe a large scale study of planners and their performance: 28 planners on 4726 benchmark problems. In the first part of the paper, we apply off-the-shelf machine learning techniques to learn models of the planners' performance from the data. In the evaluation of these models, we address the critical question of whether accurate models can be learned from easily extractable problem features. In the second part, we show how the models can be useful to furthering planner performance and understanding. We offer two contributions: 1) We demonstrate that accurate models of runtime and probability of success can be learned using off-the-shelf machine learning techniques, and 2) We show that the learned models can be leveraged to support a planner portfolio which improves over individual planners. We also discuss how the models can be analyzed to better understand how planner design decisions contribute to their performance.

Performance Models

Due to the maturity of the Planning field and the International Planning Competition (IPC), a great many planners and problems are publicly available. We are at a point where we can potentially model planner performance to better understand why they work well (when they do) and to leverage this knowledge into the next generation.

In this paper, we describe a large study of 28 publicly available planners on 4726 benchmark problems. While not complete, the planner set comprises a good cross-section of planners in terms of age and technology for planning. The benchmark problems are mostly from the IPC test set, but also include problems from other sites.

These data present several opportunities. First, planners tend to be fairly complex software systems developed for research purposes generally. Given that we have a fair amount of performance data, can accurate models of planner performance be learned? Second, can the data and models be used to improve planner performance?

We can show that models of planner performance based only on simple easy-to-compute problem features can be surprisingly accurate for predicting success and somewhat accurate (i.e., better than guessing the mean time) for predicting time. The models also work well even when trained on older problems and tested on the most recent (showing that the models generalize). We also show how the models can be embedded into a portfolio of planning systems and show that the learned models can drive a successful portfolio. Our best portfolio that uses learned models can improve over any single planner and the mean solution times. We end by discussing how the learned models can help improve our understanding of planner performance.

Related Work

Research on applying learning and planning has a long history. Our work most closely follows other research that used learning to enhance a portfolio of algorithms. Most portfolios follow one of two approaches: 1) learning models (off-line or on-line) to predict runtime/success; or 2) learning control rules to manage a single algorithm[1].

One of the first formulations calculated the risk of selecting an algorithm (Huberman, Lukose, & Hogg 1997). Gomes and Selman (1997) explicitly computed the value of multiple restarts of the same algorithm. For classical planning, the Bus meta-planner (Howe *et al.* 1999) used linear regression models of computation time and success to rank a set of classical planners.

Some researchers use a different approach wherein they learn control rules for a single overall system rather than algorithm selection among full implementations of distinct algorithms. Some early work was done by Minton (1996) for the PRODIGY system as well as Gratch and Chien (1996) for Scheduling . A more recent example is found in that of Vrakas et al. (2003) where they applied machine learning to select parameters for their planner HAP.

Predicting Planner Performance

No single planner completely dominates the others by solving all the problems that can be solved by any planner. However, a few planners – most notably those variants based on the relaxed graph plan heuristic – do solve considerably more problems. In this section, we describe how we learn

[*] This research was partially sponsored by the National Science Foundation under grant number IIS-0138690 and Air Force Office of Scientific Research grant number FA9550-07-1-0403.

[1] Space limitations preclude a complete exposition of the literature, but an analysis of most literature for the portfolio models is found at http://www.cs.colostate.edu/˜mroberts/publications.html.

performance models from a large set of run-time data and evaluate how well the learned models predict performance.

Performance Data

We ran all planners on all problems to collect performance data, using a time cutoff of 30 minutes and a memory cutoff of 768 MB. We used identically configured Pentium 4 3.4Ghz computers each with 1 Gigabyte of memory running Fedora Core 6.0 Linux.

Planners The 28 publicly available STRIPS capable planners used in our study are listed in Table 1[2]. The set is composed of IPC competitors plus some other planners included to diversify the representative approaches. The planners were all run using their default parameters.

Planner	Authors [Date]
AltAlt-1.0	Nguyen,Kambhampati, Nigenda [2002]
BlkBox-4.2	Kautz, Selman [1999]
CPT-1.0	Vidal, Geffner [2004]
FF-2.3	Hoffmann [2001]
HSP-2.0	Bonet, Geffner [2001]
HSP-h1plus	Bonet, Geffner [2001]
HSP-h2max	Bonet, Geffner [2001]
IPP-4.0	Koehler, et al. [1997]
IPP-4.1	Koehler, et al. [1997]
LPG-1.1	Gerevini, Saetti, Serina [2002]
LPG-1.2	Gerevini, Saetti, Serina [2003]
LPG-TD	Gerevini, Saetti, Serina [2005]
Metric-FF	Hoffmann [2003]
MIPS-3	Edelkamp [2003]
OPTOP	McDermott [2005]
PROD-4.0	Veloso et al. [1995]
SystemR	Lin [2001]
SAPA-2	Do, Kambhampati [2003]
Satplan04	Kautz, Selman [1999]
Satplan06	Kautz, Selman, Hoffmann [2006]
SGP-1.0h	Weld, Anderson, Smith [1998]
SGP-1.0b	Weld, Anderson, Smith [1998]
SGPlan-06	Chen, Hsu, Wah [2004]
SimPlan-2.0	Sapena, Onaindia [2002]
SNLP-1.0	McAllester, Rosenblitt [1991]
STAN-4	Long, Fox [1999]
UCPOP-4.1	Penberthy, Weld [1992]
VHPOP-2.2	Younes, Simmons [2003]

Table 1: Planners, their authors and dates of publications.

Problems Our problem collection consists of 4726 STRIPS PDDL problems from 385 domains. They are taken from Hoffmann's dataset (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 domains (Sodor and Stek) that have been made publicly available.

The majority of the problems in this compendium are outdated and no longer challenging. 33.7% of the runs succeed over all planners and problems. The mean time to completion is 19.14 seconds over all successful runs and 232.30 for failed runs. Across all planners, 82.9% (successful) and 70.6% (failure) of runs complete in under 1 second. For example, a model that predicted "success" for FF-2.3 would be correct for 68% of the problems, and a time model that predicted less than 1 second would be correct 86.9%.

The distributions do vary across planners from a range of 7.5%-70.7% for success within 30 minutes and 4.9%-70.1% for success within 1 second. The mean time to complete varies from 0.49-651.18 seconds for the different planners, with means for successful runs of 0.10-84.48 seconds and failed runs of 0.58-834.50 seconds.

Consequently, we restrict our further study to just challenging problems. A problem is defined to be *challenging* if 1) it can be solved by only three, two or one planners or 2) the median time for solution is greater than one second. We picked these thresholds so that we still had enough data to support learning/testing. These criteria reduce the set to just 1215 problems from 41 domains; all of which are solvable by some planner.

Model Parameters

We learned models for each of the planners from observed performance on a large set of benchmark problems. The input to each learning algorithm was problem/domain features and the performance (whether or not successful and how much time to completion) for each of the problems.

Performance Metrics For each planner on each problem, we recorded whether a plan was found (success as true or false) and how much time was required to complete execution (time in seconds). Because each planner has its own way of declaring success, we constructed code to automatically extract these metrics from the output. Some planners reported success in their output without actually generating a legal plan; those cases were marked as failures.

Problem/Domain Features Each problem instance is defined by 32 features that can be automatically extracted from PDDL problem and domain definitions. We started with features from (Howe *et al.* 1999) and (Hoffmann 2001), but we found that the Howe et al. features were not sufficient and the Hoffmann features, while powerful, were only tractable for small problems. The Hoffmann features were removed, and others were added based on our intuitions about what might influence performance. Table 2 shows the features in two categories: domain specific and instance specific.

Learning Planner Models

For each planner, we constructed two models: success and time. For success, we built a binary classifier (successful or not) and in some cases, used the resulting model to estimate $P(\text{solution found}|\text{problem, planner})$. Time predicts computation time needed for a given planner to successfully solve a given problem.

[2] The full references and URLS for obtaining the planners are being made available at `http://www.cs.colostate.edu/meps/nsf-data.html`.

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
boolean	requires ADL
boolean	requires conditional effects
boolean	requires derived predicates
boolean	requires disjunctive preconditions
boolean	requires domain axioms
boolean	requires equality
boolean	requires existential preconditions
boolean	requires fluents
boolean	requires quantified preconditions
boolean	requires safety constraints
boolean	requires STRIPS
boolean	requires typing
boolean	requires universal preconditions
num	# of goals
num	# of objects
num	# of inits

Table 2: The feature set: first column is the metric being collected, the last column briefly describes the feature.

We use the WEKA data mining package (Witten & Frank 2005) to build the models. We tried 32 different models from WEKA. Table 3 lists the models that were most accurate for the test used to select models for the portfolio: ADTree (ADT), DecisionTable (DT), GaussianProcess (GP), IB1, J48, JRip, KStar, LMT, Logistic (Log), MultilayerPerceptron (MLP), NNge, and PART. They include lazy instance learners, rule based, decision tree, model tree, nearest neighbor, kernel, neural network and regression algorithms.

Evaluating the Planner Models

A key issue is how well performance can be predicted given the problem characteristics. Given our methodology of using easily extracted features and off-the-shelf learning, it is entirely possible that accurate models would be elusive.

Selecting the Most Accurate Models To evaluate the models, we hold out 20% of the Challenge problems as test problems (half of IPC4 and IPC5 problems in the set); we use the remaining 80% to train classifiers. Table 3 shows the results of 10-fold cross-validation on the 80% training set (labeled "80% Challenge").

The models of success are quite accurate. Guessing "fail" can be viewed as a strawman model (it is the most likely outcome for all but two planners), and then % fail is its % correct. The best classifier's % correct is higher than (or equal to for CPT-1.0, SNLP-1.0, and VHPOP-2.2) the strawman. In some cases, the % correct is considerably higher than the strawman, e.g., FF-2.3, IPP-4.1, IPP-4.0, LPG-TD, Metric-FF, and SGPlan-06. A one-tailed paired sample t-test comparing the correctness of the strawman against the best model showed that that strawman's correctness was significantly lower ($t = -3.7, p < 0.0005$). Moreover, the average correctness is quite high: 96.59% over all planners. Which classifier is best varies across the planner set. Picking the best classifier does matter as the average accuracy across all classifiers and planners was lower (93.95%).

The models of time are not so accurate. The best Root Mean Squared Error (RMSE) averages somewhat less than the average time to succeed. The highest RMSE is 424.95 seconds, which at least is considerably less than the highest possible time of 30 minutes. However, not all planners are so hard to predict. The lowest RMSE is 3.55, and 13 have RMSEs lower than the average times. We gauge the similarity between the actual and predicted times on the test set for the challenge problems (the 20% held out) by running paired sample t-tests. At least three problems from the test set could be solved by 25 of the planners (minimum needed to test time to success predictions). Of those, predicted times for seven planners (BlkBox-4.2, FF-2.3, HSP-h2max, IPP-4.1, Satplan04, SGP-1.0b, and SGP-1.0h) were statistically significantly different at $\alpha < .05$ level; another three (HSP-2.0, IPP-4.0, and LPG-1.2) up to 0.1 and three more (CPT-1.0, OPTOP and VHPOP-2.2) up to .2. So roughly the other half had predictions similar to actual distributions.

We point out two reasons why it is harder to predict time. First, WEKA includes fewer models that can handle continuous classes; future work will look at building better machine learning models for these data. Second, the distributions are highly skewed (consider the large difference between the median and the mean) with long tails. High values of time to succeed are rare and so are very hard to predict accurately.

Do the models generalize? (Train on Old, Test on New) We trained models on the portion of the challenge data other than the latest (91 problems from the IPC5 subset) and tested on these newer problems. As Table 3 shows, the planner failure rates are comparable on the newer problems; the average accuracy of predicting success suffered only slightly (from 96.59 to 94.47).

Many of the planners are unable to solve problems from the IPC5 set. We did not calculate models of time to success for these. However, for those that could, we found lower average time to succeed and lower RMSE for the models than for the 80% Challenge set. However, the RMSE was slightly higher than the mean time. As with the 80% data, we compared actual and predicted times using two sample t-tests on the 14 planner models with sufficient data. Two (FF-2.3 and LPG-TD) were significantly different at $\alpha <$ .05, and one more (Metric-FF) at 0.2. Thus, generally the predictions were similar.

Computation Required The time to compute the models must be included in run times for on-line usage of the learned models. Fortunately, the features are fast to extract, and WEKA is fast to run the learned models. Features can be extracted from problems in 0.0025 seconds on average. The aggregated time to compute predictions for all the problems in our 80/20 test set was 0.29 for success and 0.08 for

	Success Models					Time Models					
	80% Challenge			Test IPC5		80% Challenge				Test IPC5	
Planner	% fail	Best % correct	Best Classifier	% fail	Best % correct	μ time	median time	Best RMSE	Best Classifier	μ time	Best RMSE
AltAlt-1.0	98.87	99.18	KStar	100	100	0.89	0	3.55	KStar		
BlkBox-4.2	95.77	97.42	MLP	97.8	100	11.36	0	84.79	GP	4.86	53.83
CPT-1.0	99.38	99.38	MLP	100	100	15.74	0.04	83.52	KStar		
FF-2.3	43.71	93.4	KStar	37.36	82.42	146.29	0.12	316.79	KStar	145.91	258.87
HSP-2.0	82.47	92.78	j48	94.51	94.51	552.99	0.28	421.66	KStar	46.78	58.49
HSP-h1plus	93.81	96.7	KStar	90.11	91.21	211.48	0.03	333.09	KStar	110.83	281.37
HSP-h2max	97.84	98.76	KStar	100	100	413.07	0.03	368.75	KStar		
IPP-4.0	79.28	96.08	NNge	96.7	96.7	513.88	6.29	424.95	KStar	59.74	57.73
IPP-4.1	78.45	96.39	JRip	96.7	97.8	503.23	6.45	383.94	KStar	88.49	72.02
LPG-1.1	85.46	95.36	PART	84.62	84.62	243.8	0	251.64	KStar	134.04	205.58
LPG-1.2	79.69	94.12	KStar	79.12	89.01	220.88	0	269.97	KStar	170.35	287.2
LPG-TD	63.20	94.95	IB1	37.36	71.43	214.61	0.18	309.4	KStar	138.11	190.93
Metric-FF	46.39	93.81	j48	32.97	81.32	57.26	0.54	189.98	KStar	164.42	319.6
MIPS-3	90.82	95.46	NNge	91.21	98.9	445.82	0.01	373.75	KStar	285.43	128.24
OPTOP	92.89	96.8	KStar	97.8	97.8	25.42	14	49.04	DT	193.97	154.16
PROD-4.0	98.14	99.59	NNge	100	100	588.3	0.85	170.44	KStar		
SystemR	90.31	98.97	LMT	86.81	89.1	497.41	0.14	216.14	KStar	21.52	12.84
SAPA-2	98.87	99.07	Log	100	100	302.5	0.09	250.67	KStar		
Satplan04	94.02	96.29	LMT	95.6	97.8	74.09	0.1	152.63	KStar	308.51	225.72
Satplan06	89.79	94.64	IB1	81.32	92.31	54.38	0.01	191.16	KStar	75.16	97.81
SGP-1.0b	97.84	98.76	Log	97.8	97.8	522.21	4.85	253.23	KStar	15.32	62.17
SGP-1.0h	97.84	98.87	PART	97.8	97.8	520.63	4.92	265.52	KStar	15.34	25.36
SGPlan-06	55.36	90.62	PART	24.18	87.91	113.28	0.16	294.45	KStar	14.18	32.92
SimPlan-2.0	80.82	90.93	PART	92.31	96.7	49.82	0.48	145.23	KStar	120.07	118.17
SNLP-1.0	99.59	99.59	MLP	100	100	749.16	0.1	134.45	DT		
STAN-4	98.25	98.66	ADT	100	100	11.54	0	128.98	GP		
UCPOP-4.1	96.91	98.66	LMT	100	100	842.45	8.77	235.62	DT		
VHPOP-2.2	99.28	99.28	MLP	100	100	32.24	26.02	22.49	KStar		
μ	86.61	96.59		86.15	94.47	283.38	2.66	225.92		111.21	139.11

Table 3: Planner performance and model accuracy data for success and time required using 10-fold cross-validation on 80% of the Challenge dataset as well as on the "train with old challenge problems, test on IPC5". % fail and times are from the planner performance data; "Best % correct" and "Best RMSE" measure accuracy for the models selected as best. Boldface planners are in the portfolio.

time. The mean time for WEKA to learn the models is only 4.3 seconds.

Using the Models I: A Portfolio

One obvious application of models of performance is driving an algorithm portfolio. In this case, the portfolio differs from much of the literature because the "algorithms" are actually *systems*. Consequently, we did not know what strategy to use or whether any strategy would produce performance commensurate with single planners. This section describes a portfolio and its evaluation. The evaluation is designed to answer two questions concerning the utility of the learned models. Do the learned models support the portfolio? How does the portfolio compare to single planners?

Architecture

Figure 1 shows the portfolio architecture. The previous section described the performance models. The other three decision making components can be configured into many different combinations. The components and their options are:

Algorithm selection restricts the set of possible planners to only those needed to cover the problem set. Redundant and overlapping performance in the full set of 28 planners is to be expected since we have included multiple versions of some planners and because several planners use similar technology (such as graphplan, POCL, etc.). Given the 80% challenge data, we selected a "cover" from the 28 planners using a greedy set covering approximation algorithm (Cormen *et al.* 2003). We excluded from the final list any planner that only solved one unique problem in the training set (4 planners total). Table 3 lists the cover planners in boldface.

Algorithm ranking orders the execution of the planners. The four ranking strategies are:

cover uses the set covering order,

pSuccess prunes those planners predicted to fail and orders the rest by decreasing predicted probability of success,

predTime orders by increasing predicted time, and

SimonKadane orders in decreasing $\frac{pSuccess}{predTime}$; it was shown to minimize expected cost of success in serial search (Simon & Kadane 1975).

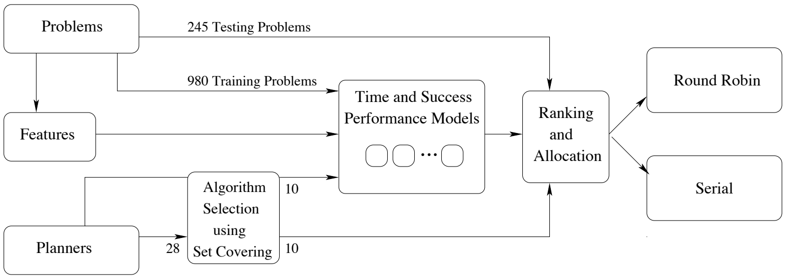

Figure 1: Portfolio architecture

Allocation strategies determine the runtime for each planner. Planners may be run serially or round-robin. The three strategies from these two types are in italics.

Serial executes the planners to their maximum allotted time and quits at the first success or after all planners have spent their time.

- ***avgPlanner*** uses the average time to succeed for the planner, and
- ***predTime*** computes the predicted time for the problem from the model.

Round-robin iterates through the queue until a single planner indicates success, all planners indicate failure, or the portfolio exceeds the experimental time limit (30 minutes), as in:

- ***confInt*** orders the CPU time of successful training runs then performs quantile analysis for each quantile q, where $q = \{25, 50, 75, 80, 85, 90, 95, 97, 99\}$. For example, if the successful runtimes of a planner are $\{0.1, 0.2, 0.3, 0.4, 0.5, 10, 100, 1000\}$, then confInt will return runtimes as $\{0.28, 0.45, 32.5, 64.0, 95.5, 370.0, 685.0, 1000.0\}$.

Are the Learned Models Useful?

We assessed the impact of ranking and allocation strategies on two performance metrics: ratio of success to failure (see Table 4) and amount of time required for solution (see Table 5). The portfolio's runtime includes the time to load the problem/domain, compute the problem features and to run the models that are used. The time to run each model (two per planner) is computed from running WEKA and is stored with the model.

For ranking, three strategies (*pSuccess*, *predTime* and *SimonKadane*) exploit the learned models. Those strategies exhibit the three lowest failure counts (29, 1 and 3, respectively) with the non-learned *cover* failing 129 times. Clearly, using the models for ranking boosts the number of problems that can be solved by the portfolio.

	Fail	Succeed
predTime:avgPlanner	0	245
predTime:confInt	0	245
SimonKadane:confInt	0	245
predTime:predTime	1	244
SimonKadane:avgPlanner	1	244
SimonKadane:predTime	1	244
pSuccess:avgPlanner	5	240
cover:confInt	6	239
pSuccess:confInt	7	238
pSuccess:predTime	8	237
cover:avgPlanner	10	235
cover:predTime	12	233

Table 4: Counts of problem solution on the 20% Challenge set for each portfolio combination, ordered best to worst.

	μ	σ
SimonKadane:confInt	49.21	136.37
SimonKadane:predTime	49.95	174.88
predTime:avgPlanner	55.53	164.08
predTime:confInt	52.57	152.67
SimonKadane:avgPlanner	60.66	200.67
predTime:predTime	62.73	197.00
pSuccess:avgPlanner	91.80	282.33
pSuccess:confInt	94.21	287.95
pSuccess:predTime	101.18	308.39
cover:confInt	138.94	376.94
cover:avgPlanner	150.66	422.63
cover:predTime	153.68	431.62

Table 5: Mean and standard deviations of portfolio combination runtimes for 20% Challenge set, ordered best to worst.

The three model based ranking strategies also excel on runtime, posting the three lowest mean runtimes as shown in Table 6. *SimonKadane* ranking appears to outperform the other methods based on the means. However, we used Tukey's Honest Significant Difference method (with an adjusted significance level to control for experiment-wise error at $\alpha = 0.05$) to group performance that is not significantly different. Tukey's formed two groups: [*SimonKadane*, *predTime*, *pSuccess*] and [*pSuccess*, *cover*]. The learned models appear to exert a similar positive impact on ranking success.

For allocation, *predTime* is the only model-based strategy. On failure count, it performed worse (52) than *avgPlanner* (35), but better than *confInt* (77). On runtime, *predTime* is again third best. TukeyHSD grouped [*avgPlanner*, *predTime*]. These results suggest that allocation is not well served by the time model in the current strategies. However, the effect is mitigated by an interaction effect; the lowest mean runtime is for *SimonKadane:confInt* which uses

Strategy	μ	σ
SimonKadane	53.43	173.05
predTime	58.11	177.28
pSuccess	96.25	294.40
cover	149.11	415.51
avgPlanner	110.94	342.21
predTime	128.94	381.97
confInt	185.58	459.44

Table 6: Grouped means and σs for runtimes for ranking and allocation strategies.

the supposed worst allocation strategy and the second lowest was *SimonKadane:predTime* which uses models for both ranking and allocation.

The best portfolio (*SimonKadane:confInt*) uses a learned model for ranking but not allocation, reflecting the lower accuracy of the time prediction. We compared the best portfolio to single planners (those in the cover set). As Table 7 shows, the most successful portfolio clearly outperforms any single planner on the testing set. Comparing the runtime of the portfolio to that required by the single planner that solved the most problems in the test set (SGPlan-06) shows that *SimonKadane:confInt* is better, but not significantly ($t = -1.497$, $p < 0.136$ on a paired sample t-test of problems solved by both). SGPlan-06 takes an average of 41.2 seconds; the portfolio took 32.61 seconds.

	FAIL	SUCCEED
SimonKadane:confInt	0	244
sgplan-2006	51	194
lpg-td-1.0	108	137
metric-ff-2002	174	71
ff-2.3	174	71
blackbox-4.2	213	32
lpg-1.2	225	20
ipp-4.0	243	2
optop-1.6.19	244	1
vhpop-2.2	245	0
prodigy-4.0	245	0
ucpop-4.1	245	0

Table 7: Comparison of best portfolio and single planners on problems solved.

Using the Models II: Understanding Performance

We have just started examining the models for what they can tell us about the planners. For example in a paper submitted to the ICAPS07 workshop on Domain Heuristics, we have shown that features from Hoffmann's taxonomy distinguish performance of h+ planners from those that do not use h+.

The most exciting future work is in deconstructing the portfolio and the models. We now have a tremendous amount of data on planner performance. We plan to examine the data for interactions based on planner type (such as SAT-based, POCL-base, Graphplan, Relaxed Graphplan, or Hybrid). Our study used the default settings of all planners; we have already began to extend these results to include some limited parameter tuning of the planners by viewing each change in parameters as a new planner. Most importantly, we plan to analyze the learned models for clues as to why some planners do better on some types of problems.

Our study has demonstrated the feasibility of modeling planner performance using simple features and off-the-shelf machine learning methods. The learned models have been effective at predicting whether or not a planner is likely to succeed on a never before seen problem. The learned models, so far, are not so good at predicting how long a given planner will take on the problems. The learned models were also useful in driving a portfolio of planners to outperform any single planner.

References

Cormen, T.; Leiserson, C.; Rivest, R.; and Stein, C. 2003. *Introduction to Algorithms*. MIT press, Cambridge, MA.

Gomes, C. P., and Selman, B. 1997. Algorithm portfolio design: Theory vs. practice. In *Proc. of 13th UAI*. Linz, Austria.: Morgan Kaufman.

Gratch, J., and Chien, S. 1996. Adaptive problem-solving for large-scale scheduling problems: A case study. *Journal of Artificial Intelligence Research* 4:365–396.

Hoffmann, J. 2001. Local search topology in planning benchmarks: An empirical analysis. In *Proc. of 17th IJCAI*, 453–458.

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

Howe, A. E.; Dahlman, E.; Hansen, C.; von Mayrhauser, A.; and Scheetz, M. 1999. Exploiting competitive planner performance. In *Proc. of 5th ECP*.

Huberman, B. A.; Lukose, R. M.; and Hogg, T. 1997. An economics approach to hard combinatorial problems. *Science* 275:51–54.

Minton, S. 1996. Automatically configuring constraint satisfaction programs: A case study. *Constraints* 1(1/2):7–43.

Simon, H., and Kadane, J. 1975. Optimal problem-solving search: All-or-none solutions. *Artificial Intelligence* 6:235–247.

Vrakas, D.; Tsoumakas, G.; Bassiliades, N.; and Vlahavas, I. 2003. Learning rules for adaptive planning. In *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS03)*, 82–91.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical machine learning tools and techniques*. Number ISBN 0-12-088407-0. San Francisco: Morgan Kaufmann.