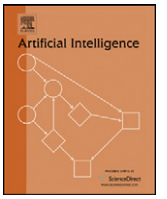
Artificial Intelligence 173 (2009) 536–561

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint



Learning from planner performance

Mark Roberts *, Adele Howe

Computer Science Department, Colorado State University, Fort Collins, CO 80523-1873, USA

ARTICLE INFO

Article history:
Received 1 November 2007
Received in revised form 10 November 2008
Accepted 10 November 2008
Available online 28 November 2008

Keywords:
Classical planning
Empirical studies of performance

ABSTRACT

The planning community has amassed a large body of publicly available problems in a standardized input language and planners that accept the language. We seized this remarkable opportunity to collect data about how some of these planners perform on the benchmark problems. We analyzed the resulting data to learn about the state of the art in Classical planning.

Our analyses are retrospective, prescriptive and prospective. The first analyses are retrospective and prescriptive in that they characterize the problems and planners in terms of difficulty, diversity and trends over time. We statistically confirm that problem sets have become more difficult and that new planners are generally more capable. A visualization of planner success on domains shows how the domains distinguish performance. We also assess whether some older planners can be disregarded as out-dated and find that while they are not up to current capabilities, some do provide limited distinct functionality.

The second analyses automatically learn models of success and time for each planner. The models are constructed from easily extracted features of problems and domains and use off-the-shelf Machine Learning techniques. We find the models of success to be extremely accurate, but the models of time to be less so. They too are both retrospective and prescriptive in demonstrating the predictability of current planner performance.

In a third analysis, we apply the data to an existing explanatory model linking the relationship between the search space and planner performance. Our study validates previous results linking search topology with planner performance on a wider set of planners than the original study.

Finally, we fill in some gaps in observed performance of the benchmark problems by constructing new problems; these problems do turn out to be more challenging. This study of existing and new problems and planners is prescriptive and prospective in that the results should help guide researchers in comparatively evaluating their planners and suggest need for additional effort.

These analyses highlight the importance of problems in driving research in planning. We show how much can be accomplished with the available resources and point out how much more can be done by broadening the problems available and by learning from what has already been done.

© 2008 Elsevier B.V. All rights reserved.

* Corresponding author.
E-mail addresses: mroberts@cs.colostate.edu (M. Roberts), howe@cs.colostate.edu (A. Howe).
URLs: http://www.cs.colostate.edu/~mroberts (M. Roberts), http://www.cs.colostate.edu/~howe (A. Howe).

0004-3702/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2008.11.009

1. Assessing planning

Developing a state of the art planning system requires tremendous effort. The systems are large – MetricFF contains more than 29,000 lines of code – with each new planner introducing a set of innovations. Sometimes the innovations are minor in that they simply provide the next step in the development of an existing system. Other innovations branch into an entirely new paradigm, such as the introduction of the planning graph in Graphplan [1].

Remarkably, more than 30 "classical" planning systems are publicly available. Also, remarkably, these systems accept essentially the same input language: a version of Planning Domain Definition Language (PDDL) or a syntactic variant. To complement the systems, over 5000 STRIPS [2] and ADL [3] planning problems are publicly available in PDDL. Thus, performance comparison is well supported by the community.

The availability of so many systems and problems is due in large measure to the International Planning Competitions (IPCs). The first competition, run by Drew McDermott, set the framework and defined the first version of the input language, PDDL [4]. Subsequent IPCs extended PDDL and significantly broadened the problem set and types of planners involved.

The IPCs have also encouraged assessments of the state of the art in the field through comparisons of performance of the participants and archival publications of the best systems (a special issue and a special track in *Journal of Artificial Intelligence Research*). Organizers of several competitions undertook extensive analyses of performance (e.g., [5,6]). These efforts have contributed to the dramatic improvement that planning systems have exhibited during the period, as evidenced by the problems in subsequent IPCs.

The IPCs have shown how difficult it is to declare a winner; many factors have to be balanced (e.g., the quality of plans produced, the amount of time taken to solve a problem, whether the planner was designed to solve particular types of problems, the language features handled, etc.). Competitors often solve different numbers and types of problems. While single planners sometimes significantly outperform the others on the metrics, often no single planner completely dominates, and older planners may still offer decent performance. For example, the organizers of IPC4, Hoffmann and Edelkamp, included the most successful IPC3 planner, LPG-1.2, in their analyses and found that it performed in the middle of the pack in IPC4 [5].

To complement the analyses of the IPCs, we conducted a retrospective assessment of planning *across* the IPC problems and planners. We ran 28 planners on 4726 problems using identically configured hardware and collected basic performance information.[1] We limited our study to STRIPS and ADL problems because they represent the core capability of planning systems and provided the greatest coverage (most planners and problems). While "classical" planners are severely limited in what they can accomplish in terms of realistic planning problems, STRIPS/ADL planning is by far the most mature subarea with a large following; there are still very active classical tracks in the IPCs. Thus, we view our study here as foundational and encourage similar studies of more capable planners as those subareas produce a critical mass of planners and problems. Sections 2 and 3 of the paper list the planners and problems in our study; brief descriptions of planners are given in Appendix B.

The studies described in the paper have multiple goals related to analyzing and extending the state of the art. Broadly, the studies are designed:

1. to assess the difficulty of the problems and the capabilities of the current publicly available planners (in Section 4),
2. to learn models of performance (in Section 4.5) as a means of assessing how well current performance can be characterized and predicted,
3. to start to explain some observed trends in performance (in Sections 5 and 6), and
4. to develop some new challenging problems to augment the existing benchmarks (Section 6).

We hope that the results of the first will help researchers decide which problems to use and planners to compare to during their own experiments. For the second goal (in Section 4.5), we show that whether a planner is likely to successfully solve a problem can be learned from the performance data and some inexpensive problem features, but that predicting time to solution is more difficult. We show how the performance data can validate an existing model that links search space topology with planner performance (in Section 5). Our studies point out deficiencies in the current benchmark problem sets. Thus, we offer some newer problems, designed to be challenging, and show how they fit into the existing set (in Section 6).

To some extent, all the studies are retrospective in examining planners and problems developed over more than a decade. Our first, second and third studies are also prescriptive in guiding comparative evaluation of planners. Our third and fourth are prospective in starting to explore directions for much needed further research in planning: explaining planner performance and constructing new problem sets.

[1] This was a fairly complete sample as of 2006 when we began the data collection. We will make the complete data set or the scripts with which we generated and validated the data available upon request.

Table 1
Planners ordered approximately chronologically. The 'h^+' column indicates those planners that were labeled as an h^+ planner (see Section 5).

Planner	h^+	Approximate date	Search type
SNLP-1.0		1991	partial order causal link
UCPOP-4.1		1992	partial order causal link
PROD-4.0		1995	partial order causal link
BlkBox-4.2		1996	SAT/BDD
IPP-4.1		1997	GraphPlan
IPP-4.0		1997	GraphPlan
SGP-1.0b		1998	GraphPlan
SGP-1.0h		1998	GraphPlan
STAN-4		1999	GraphPlan
FF-2.3	*	2001	heuristic search
HSP-2.0	*	2001	heuristic search
HSP-2.0r-h1plus	*	2001	heuristic search
HSP-2.0r-h2max	*	2001	heuristic search
SystemR		2001	STRIPS
AltAlt-1.0	*	2002	hybrid
LPG-1.1	*	2002	heuristic search
SimPlan-2.0		2002	heuristic search
LPG-1.2	*	2003	heuristic search
Metric-FF	*	2003	heuristic search
MIPS-3		2003	SAT/BDD
SAPA-2		2003	heuristic search
VHPOP-2.2		2003	partial order causal link
SGPlan-06	*	2006	hybrid
CPT-1.0		2004	partial order causal link
FastDown	*	2004	heuristic search
LPG-TD	*	2005	heuristic search
OPTOP		2005	heuristic search
Satplan06		2006	SAT/BDD

2. The planners

Our study comprises 28 publicly available STRIPS (and some ADL) capable planners. The planner set is intended to capture, as much as possible, the broad history of classical planners, with a requirement that the planners be easily made to accept PDDL. Our effort begins with the planners from an earlier study [7], and moves forward in time collecting as many planners as possible up through IPC4. The set is composed of IPC competitors plus some other planners included to diversify the representative approaches. In some cases we have planner code for older planners that will not compile/run on newer platforms. In two cases we found reporting discrepancies with older versions of planners (Satplan04 and SGPlan-04) and decided to include the most recent version to maintain adequate representation. For some planners, we include multiple variants to help assess progress. We have obtained additional planners that we have not yet run or are waiting on authors of those systems to send code; we welcome additions to this list.

In Table 1, we list the planners chronologically. To show the diversity of approaches, the table also groups the planners into "types" by their primary search representation (such as Graphplan, Heuristic Search, Partial Order Causal Link, SAT encoding, etc.). Because of the number of planners, even a short description of every planner takes many pages; consequently, we have placed short descriptions in Appendix B.

3. The problems

Our current problem collection consists of 4726 STRIPS PDDL problems from 96 distinct domains. They are taken from Hoffmann's benchmark problem set [8], 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.

As with the planners, we seek to include as diverse and historical a set of problems as possible. We focus on STRIPS and ADL problems that are directly represented in PDDL. A problem is included if it requires none, some, or all of the ADL subset. We recognize that some planners cannot process all of the ADL, but include them as a way of distinguishing performance of planners that do handle the ADL.

We organize the problems in our study by the set from which the problem was obtained and by its domain. Tables 16 and 17 (in Appendix A) list the problem sets along with the domain names; detailed descriptions of most of these domains may be found in the original references, as follows. About half of our domains originated in the International Planning Competitions (IPCs) held biennially in conjunction with the International Conference on Automated Planning and Scheduling (ICAPS, formerly AIPS): IPC1 [9] had 7 domains, IPC2 [10] had 5 domains; IPC3 [6] had 7 domains that were within the expressive ability of STRIPS planners; IPC4 [5] had 7 such domains as did IPC5 [11]. The Strict domain collection is from the original UCPOP distribution set, while the Hoffmann problems [8] are the set of problems he collected for his empirical analyses.

4. The study: Assessing planner performance

Conventional wisdom suggests that the most recent problems are the most demanding and the most recent planners are the most effective. The increasing challenge posed by the IPCs supports the former supposition, but a previous study [12] questions the latter.

Our study collects and analyzes performance data from a wide range of problems and planners in part to examine the conventional wisdom and in part to capture insight about which planners do well when. In this section, we describe our data collection and analysis to understand the role of the problems and planners in comparative studies.

4.1. Collecting the performance data

For each planner on each problem, we record 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 bash scripts using Unix utilities (such as sed, grep, and ulimit) to automatically extract these metrics from the output. Some planners report success in their output without actually generating a legal plan; we created another set of scripts that checked for as many of these cases as possible. Aside from this process, we believe the planners when they report success. Runs that timed out are counted as failures for the purpose of this study.

The planners were all run using their default parameters on 30 identically configured workstations: HP-XW4300-P4/650 machines with 3.4 GHz Pentium processors and 1 Gb memory. The planners were allotted a maximum of 30 minutes and 768 Mb to solve the problems. Planners that used Lisp were run with Allegro Common Lisp Version 8.1. Planners that used stochastic search were allowed only a single run to keep comparisons fair in terms of CPU times. We recognize that allowing a single run may partially limit the ability of such planners, but felt that the limitation was not crippling (for example, the LPG family remains a strong contender despite this limitation). Additionally, any planner can apply stochastic or restart search within its allotted time (for example, as those in the SGPlan or SATPlan family do); we only stop the planner when it reports success, failure, or when it exhausts its time or memory limits.

4.2. Analysis tools

We apply a variety of statistics, statistical tests and visualizations to draw conclusions about the performance data. They are all available in the R statistical package [13]. For those unfamiliar with the tests, we offer brief descriptions here.

Sammon Map is our primary visualization technique. It is a special case of multidimensional projection of the data from n dimensions down to k dimensions while preserving as much as possible the relative distances between the high-dimensional points. Generally, it iteratively removes the least significant dimension according to some distance measure until the dimension of k is reached. The value of k and the distance measure are parameters to the search; we used $k = 2$ and the Euclidean distance. Thus, the distance matrix equates to a visualization of the data along the first two principal components where the axes of such plots are simply the scaled components. The key message from these plots is that the proximity of points indicates greater similarity. We used the function available in the MASS package of the R statistical package.

T-tests compare two (continuous) sample vectors $(\mathbf{x}, \mathbf{y})$, to determine the extent to which they are drawn from the same underlying distribution Z. Generally, this is done by comparing the means and standard deviations of the two sample distributions, assuming that they are Normal. A significant t-test indicates that $\mathbf{x}$ and $\mathbf{y}$ are very likely from different distributions. A paired sample t-test is performed over the difference vector $\mathbf{z} = \mathbf{x} - \mathbf{y}$ for each paired experimental condition; this test assumes that the distribution of Z has a mean of zero. Both versions of this test are robust (and nearly normal) for sufficiently large n. In all cases, our n is large enough to support this assumption.

Wilcoxon Signed-rank Test is a non-parametric version of the paired-sample t-test. It replaces continuous values with ranks and, as with the paired sample t-test, compares the rank of matched pairs of data. It is often used as a substitute for the t-test when the data are not Normally distributed.

Tukey's Honest Significant Difference (TukeyHSD) Test can handle multiple pair-wise tests while still controlling the overall experimental error rate. Tukey's HSD test looks at the number of steps between equal groups means when they are ordered from smallest to largest.

ANalysis Of VAriance (ANOVA) compares several groups of observations to determine whether the means across the groups all appear equal. If the F value (the test statistic computed from comparing the groups) is significant, then we have shown a main effect: the value of the observation is said to depend on the group to which it belongs. A significant ANOVA justifies further pair-wise tests.

Chi-squared Test compares sets of frequencies under different values of an independent variable to determine whether the proportions are equal across the values. In our usage of the test, the test statistic (χ^2) is calculated from a contingency table, in which the columns are the different conditions/samples, the rows are the different values of a discrete independent variable and the cells are the observed frequencies.

G-Test or Log-Likelihood Ratio Test performs the same kind of analysis as the chi-squared test. However, it is a more accurate version of the χ^2 statistic, which was an approximation introduced to avoid hand-calculating log-likelihood values. While the G-test is being used more frequently, it has the disadvantage that it cannot be used when any cell value is zero.

Generally, we use t-tests and ANOVAs when the dependent variable is continuous and chi-squared tests or G-tests when the dependent variable has a small number of discrete values. We use Wilcoxon and Tukey's to improve robustness of observations when we question the assumption of normality or are performing a large set of paired comparisons. We recognize that some tests have strong assumptions about normality, though it is standard practice to use these tests in the case of sufficiently large samples.

4.3. Observations about the problems

The problems in our set were developed over a 15 year period. Versions of the problems continue to be used today when small problems are needed for proof of concept or theoretical analyses (e.g., the pertinacious Sussman's Anomaly problem). Many of these may be outdated and no longer challenging. The more recent problems are designed to challenge the recent IPC contenders or to showcase particular planner capabilities. Because of the large number of problems available, researchers cannot possibly (and indeed should not) report results on them all. We examine four factors that help define the utility of the problems for experimentation.

4.3.1. How difficult are the domains/problems to solve?

As a complete set, the answer is "not very". We consider difficulty as measured by how many planners can successfully solve problems and how long it takes them to do so. 33.8% of the runs succeed over all planners and problems. The mean time to completion is 14.99 seconds over all successful runs and 263.30 seconds for failed runs. Across all planners, 81.9% successful and 68.1% failed runs complete in under one second.

Table 16 organizes the success statistics by the domains. The second column is the number of problems in that domain, and third is the number of those problems that can be solved by at least half of the planners. The next three columns are the ratio of runs that succeed within 30 minutes, within 10 seconds, and within 1 second. The last column counts the number of planners that can succeed on at least one problem from the domain.

Many of the domains are simple. For 25 domains (with a total of 962 problems), all of their problems can be solved by at least half of the planners (Table 16 'Half' column, in bold). An additional 4 domains (with 158 problems) had a success rate over all runs that exceeded 50% (Table 16 'Suc' column, in bold). All but one of these were from the either intentionally small Hoffmann data set or the old Strict dataset, which supports the hypothesis that many of the older problems are indeed out-dated. On the other hand, 538 problems were not solved by any planner. As shown in Table 2 (Total minus Solved), 422 of those problems are from the IPC3, IPC4 and IPC5 sets, though 10 were from IPC1, 12 from IPC2, 82 from Strict, and 12 from Stek.

Another factor for problem difficulty is how much time is required to solve the problems. In Table 16, the three columns related to time to successful completion show little difference in success ratio when more time is allocated. Only 25 domains show an increase of more than 0.05 success ratio[2] when increasing time from 10 seconds to 30 minutes (Table 16 '10Sec' column, in bold); an additional 17 domains (for at total of 42) show an increase of 0.05 success ratio when increasing time from 1 second to 30 minutes (Table 16 '1 Sec' column, in bold). Table 17 shows time required to finish, successfully solve, and fail to solve for the domains. Little time is required for most successful runs (scan "med" column under "Successes"). Only 15 domains had a median time to success of over one second, and only 10 domains had a mean time to success of over 60 seconds. Thus, if problems are solved, they are generally solved very quickly. Failures can take considerably longer, which is due to the tenacity of some planners.

4.3.2. How well do the problems distinguish performance?

As with problem difficulty, we have identified several ways of assessing how problems distinguish performance. First, we look at how many problems could be solved by only a single planner, as shown in Table 3. There were 239 such problems in 21 domains. These problems are either especially difficult or favor a particular approach.

Interestingly, two of the oldest planners, UCPOP-4.1(UC) and PROD-4.0, manage to solve a handful of problems not solved by the other planners. All of these problems are from the Strict set. UCPOP's formulation includes search control for the simple domain (which is a version of truckworld). The eight-puzzle and the Stek domains (both move tiles/tapes subject to certain physical and resource constraints) may be well suited to Prodigy's style of means-ends analysis.

Second, we construct a Sammon map across the domains. The domain vector had one position per planner with a count of how many problems from the domain were solved by that planner normalized to a ratio; this was a 28-dimensional space before applying multidimensional scaling. So if two domains are equally difficult for every planner, they will have similar vectors.

[2] 5% is an arbitrary value to show our point.

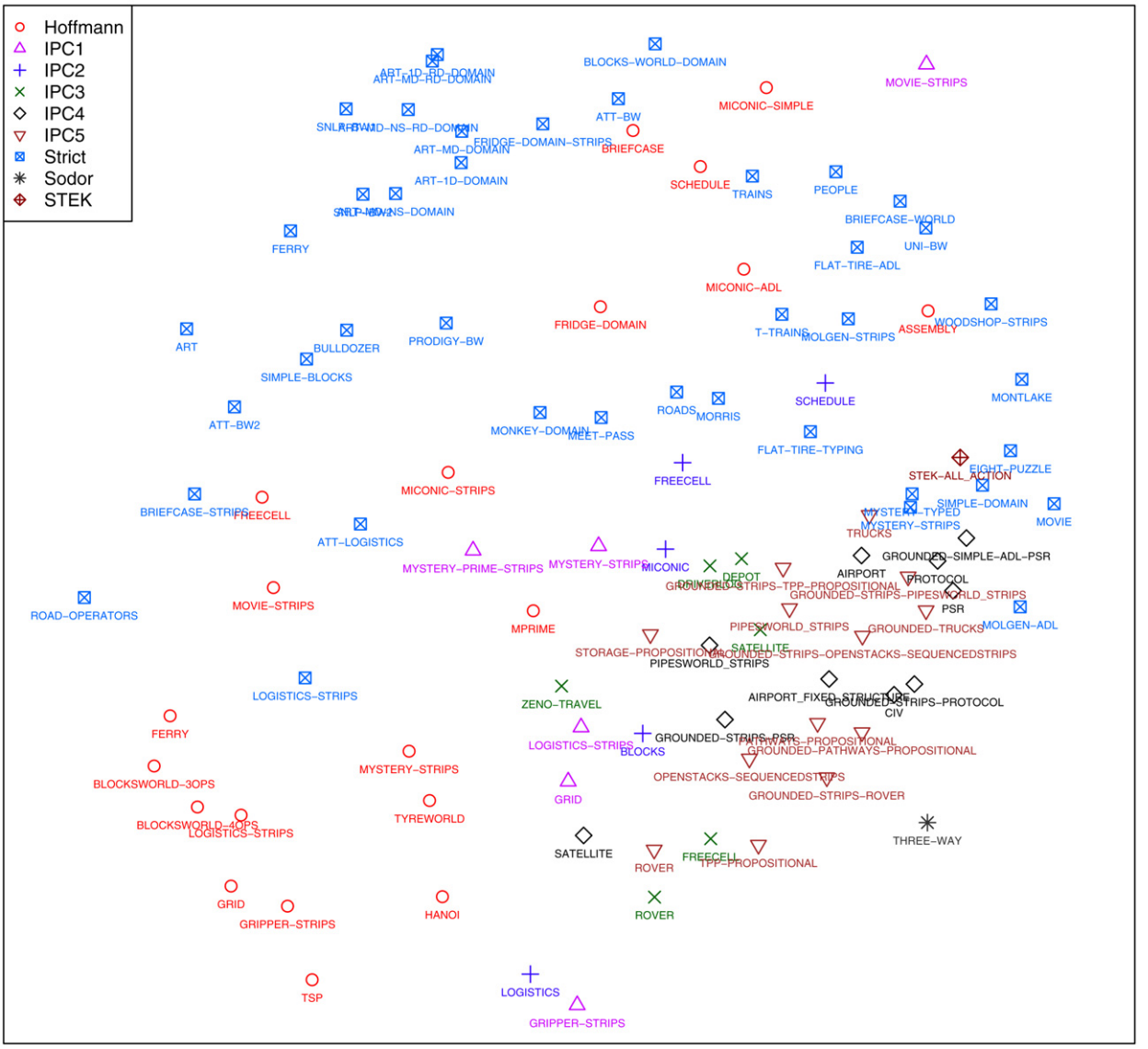


Fig. 1. Sammon map of domains. Domain points are colored and of a particular shape based on their membership in a problem set. Prior to multidimensional scaling, each domain consisted of a 28-dimensional vector of the number of problems solved by each planner normalized to a ratio.

Fig. 1 shows the map that results; each problem set is colored and shaped differently. This map illustrates how similar domains are with respect to how well the planners do on them. For example, many of the IPC5 domains are drawn together on the bottom right; few planners solve these domains and those that do tend to do so across the domains in the problem set. Based on the proximity of the later IPC points (IPC3, IPC4, IPC5) as compared to other more dispersed problem sets, it appears that these domains play a similar role in differentiating planner performance. Different versions of domains (e.g., logistics-strips from Strict and IPC2, satellite from IPC3 and IPC4) tend to be proximal. The most dispersed problem sets are Strict and Hoffmann probably because more different planners can solve these domains and because they came from a variety of sources originally.

4.3.3. Have the problem sets become more challenging over time?

Taken across all planners, we can show that the problem sets have indeed become increasingly challenging. We assess the trend according to each performance metric: success and time to solution. For success, we construct two contingency tables (shown in Table 2) comparing rate of problems solved overall and rate of problems solved by half the planners across the IPC problem sets. Pearson's chi-square tests on each are significant for all problems ($\chi^2 \approx 19.58$, $p < .001$) and those solved by half ($\chi^2 \approx 100.97$, $p \ll 0.0001$), which supports the hypothesis that the rates differ across the problem sets. The test does not address whether the rate decreases over time, but the table clearly shows that it does with the ratio of

Table 2
Contingency tables for rate of success across all planner runs and problems and rate of problems solved by half the planners. "Solved" is the total number of problems solved across all planners; "Solved By Half" is the number of problems solved by at least half the planners.

	Solved	Total	Ratio
IPC1	145	155	0.94
IPC2	633	645	0.98
IPC3	355	440	0.81
IPC4	606	858	0.71
IPC5	365	450	0.81
Total	2104	2548	

	Solved By Half	Total	Ratio
IPC1	50	155	0.30
IPC2	88	645	0.11
IPC3	84	440	0.17
IPC4	52	858	0.05
IPC5	22	450	0.04
Total	296	2548	

Table 3
Problems uniquely solved by a single planner, organized by problem set and those planners that uniquely solved at least one problem. Planner names are abbreviated to save space in the table: FD = FastDown, UC = UCPOP-4.1, VH = VHPOP-2.2. Planners that did not uniquely solve a problem are excluded from this table.

Domains	CPT	FD	FF	LPG1.2	LPG-TD	M-FF	OPTOP	Prodigy	R11	SGPLAN	UC	VH	total
IPC1													
mystery	1	0	0	0	0	0	0	0	0	0	0	0	1
IPC2													
miconic	0	0	0	0	0	1	3	0	0	0	0	0	4
schedule	0	0	2	0	0	0	0	0	0	0	0	0	2
IPC3													
depot	0	0	0	0	6	0	0	0	0	0	0	0	6
driverlog	0	0	0	0	5	0	0	0	0	0	0	0	5
zeno-travel	0	0	6	0	0	0	0	0	0	0	0	0	6
IPC4													
airport	0	0	0	0	11	0	0	0	0	8	0	0	19
civ	0	0	0	0	0	0	0	0	0	1	0	0	1
pipesworld	0	3	0	1	1	1	0	0	0	7	0	0	13
promela-phil	0	18	0	0	0	0	0	0	0	11	0	0	29
promela-opt	0	2	0	0	0	0	0	0	0	25	0	0	27
psr	0	72	5	0	0	0	0	0	0	4	0	0	81
IPC5													
openstacks	0	0	0	0	0	0	0	0	0	1	0	0	1
pipesworld	0	4	0	0	3	1	0	0	0	6	0	0	14
rovers	0	9	0	0	0	0	0	0	0	0	0	0	9
storage-prop	0	0	0	0	6	0	0	0	0	0	0	0	6
trucks	0	2	0	0	0	0	0	0	0	1	0	0	3
Hoffmann													0
Strict													
eight-puzzle	0	0	0	0	0	0	0	2	0	0	0	0	2
molgen	0	0	0	0	0	0	0	0	0	0	0	1	1
simple	0	0	0	0	0	0	0	0	0	0	4	0	4
trains	0	0	0	0	0	0	0	0	1	0	0	0	1
STEK	0	0	0	0	0	0	0	4	0	0	0	0	4
total	1	110	13	1	32	3	3	6	1	64	4	1	239

successful runs ranging from 0.94 to 0.71 and the ratio of problems solved by half declining from 0.30 to 0.04 between IPC1 and IPC5.

For time, we first run a one-way ANOVA comparing time to success for each of the problem sets. As with the χ^2, these results are also highly significant ($F \approx 28.374$, $p \ll 0.0001$), suggesting that time to succeed depends on problem set. To test whether the problem sets got successively harder, we compare successive pairs using a two sample t-test. The only pairing that showed a significant difference was IPC2 against IPC3 ($t \approx -5.38$, $p \ll 0.0001$). Indeed, the mean times for each data set show a jump between those two, but not much difference otherwise (means of 30.58, 25.11, 42.60, 41.81 and 45.98 for IPC1, IPC2, IPC3, IPC4 and IPC5, respectively).

4.3.4. Are some problems more challenging?

Because of the large number of small, easy-to-solve problems, we attempt to focus some analyses on the more challenging problems. A problem is defined to be *challenging* if 1) it can be solved by only one, two or three planners or 2) the median time for solution is greater than one second. Admittedly, these thresholds are somewhat arbitrary as they were chosen to provide enough data to support learning/testing (about a third of the original data). These criteria reduce the set

Table 4
Counts of the challenge problems from each problem set.

Problem set	Count
Hoffmann	135
IPC1	53
IPC2	318
IPC3	164
IPC4	489
IPC5	226
Strict	26
Sodor	5
STEK	12
Total	1428

Table 5
Planner success ratio for each planner by problem set. The planners are ordered by approximate age. A zero indicates the planner had zero successes in that problem set.

Planner	Hoffmann	IPC1	IPC2	IPC3	IPC4	IPC5	Strict	Sodor	STEK
SNLP-1.0	0.31	0.09	0	0.02	0.01	0	0.22	0.38	0
UCPOP-4.1	0.44	0.3	0.04	0.03	0.03	0.03	0.62	0.23	0.33
PROD-4.0	0.40	0.21	0.02	0.07	0	0.04	0.53	0.15	0.5
BlkBox-4.2	0.55	0.25	0.11	0.16	0.14	0.15	0.17	1.00	0.17
IPP-4.0	0.84	0.39	0.22	0.19	0.08	0.12	0.67	0	0
IPP-4.1	0.97	0.39	0.28	0.16	0.08	0.12	0.67	0	0
SGP-1.0b	0.47	0.48	0.03	0.11	0.04	0.04	0.62	0.15	0
SGP-1.0h	0.47	0.48	0.03	0.11	0.04	0.04	0.62	0.15	0
STAN-4	0.37	0.52	0	0.12	0	0.02	0.09	0.38	0
FF-2.3	0.93	0.71	0.93	0.67	0.19	0.36	0.65	0	0
HSP-2.0	0.49	0.42	0.14	0.26	0.10	0.10	0.28	0.23	0
HSP-2.0r-h1plus	0.45	0.43	0.10	0.27	0.09	0.12	0.16	0.23	0
HSP-2.0r-h2max	0.44	0.26	0.08	0.10	0.07	0.07	0.15	0.23	0
SystemR	0.53	0.19	0.45	0.25	0.05	0.12	0.07	0	0
AltAlt-1.0	0.37	0.56	0	0.08	0	0.02	0.09	0.54	0
LPG-1.1	0	0.38	0.12	0.32	0.04	0.08	0.03	0	0
SimPlan-2.0	0.61	0.34	0.56	0.48	0.04	0.09	0.28	0	0
LPG-1.2	0.58	0.65	0.40	0.57	0.11	0.15	0.57	0	0
Metric-FF	0.93	0.68	0.84	0.72	0.19	0.37	0.64	0	0
MIPS-3	0.35	0.43	0.09	0.34	0.07	0.09	0	0	0
SAPA-2	0.22	0	0.28	0.18	0.06	0	0	0	0
VHPOP-2.2	0.46	0.12	0.07	0.06	0.04	0.06	0.05	0	0
SGPlan-06	0.74	0.68	0.53	0.72	0.48	0.70	0.64	0	0
CPT-1.0	0.56	0.32	0.12	0.15	0.08	0.10	0.51	0.15	0
FastDown	0.70	0.54	0.44	0.73	0.52	0.61	0.03	1.00	0
LPG-TD	0.63	0.67	0.46	0.62	0.36	0.65	0.61	0	0
OPTOP	0.35	0.03	0.27	0.18	0.08	0.16	0.16	0	0.17
Satplan06	0.46	0.39	0.17	0.31	0.21	0.30	0.48	0	0

to 1428 problems from 47 domains; all of which are solvable by some planner. Table 4 shows the counts by problem set of this subset of problems. Note that 135 problems remain from the Hoffmann set in the Assembly domain and 26 problems from the Strict set (in a variety of domains). We will use this problem set throughout the paper; we refer to it as the *Challenge* problem set.

4.4. Observations about the planners

We analyze the observed performance based on planners to assess progress in the field and to help in designing comparison experiments. In particular, a key issue is whether some of the older planners can be declared obsolete.

4.4.1. How does performance of new planners compare to old?

Several of the more recent planners (LPG-TD, FastDown, Metric-FF and SGPlan-06) exhibit high rates of success. Table 5 shows the success rate for each planner in each data set. FF has the highest rate on IPC1 and IPC2, as well as rates above 0.5 for Hoffmann, IPC3, and Strict sets. FastDown has the highest rate for IPC3 and IPC4 as well as rates above 0.5 for IPC1, IPC5, and Hoffmann. SGPlan has the highest rate for IPC5 as well as high rates across most sets. LPG-TD has high rates among most sets.

Table 6
Planner median time to success by problem set. A dash indicates the planner had zero successes in that problem set, while a zero indicates a median time of smaller than one hundredth of a second. The planners are ordered by approximate age.

Planner	Hoffmann	IPC1	IPC2	IPC3	IPC4	IPC5	Strict	Sodor	STEK
SNLP-1.0	0.18	10.9	0.29	133.78	321.48	492.61	0.06	8.01	–
UCPOP-4.1	0.45	6.88	0.3	368.1	1.31	2.78	0.18	161.64	0.54
PROD-4.0	0.56	0.6	0.57	0.66	–	0.72	0.52	0.66	2.41
BlkBox-4.2	0	0.16	0.34	5.44	10.46	1.56	0	22.89	0.2
IPP-4.0	0	0.75	0.02	0.26	0.02	0.14	0	–	–
IPP-4.1	0	0.84	0.04	0.58	0.02	0.15	0	–	–
SGP-1.0b	0.12	0.46	0.12	6.91	0.16	26.56	0.1	0.24	–
SGP-1.0h	0.12	0.46	0.12	6.86	0.16	26.62	0.1	0.23	–
STAN-4	0	0.02	–	0.1	–	0.18	0	0	–
FF-2.3	0	0.02	0.04	0.1	0.38	0.08	0	–	–
HSP-2.0	0	0.74	0.14	1.16	0.09	0.01	0	0.03	–
HSP-2.0r-h1plus	0	0.62	0.01	0.6	0.02	0.01	0	0.03	–
HSP-2.0r-h2max	0	0.88	0.01	0.08	0	0	0	0.03	–
SystemR	0.05	0.13	0.34	0.66	0.19	0.66	0.05	–	–
AltAlt-1.0	0	0.03	–	0.07	–	0	0	0.02	–
LPG-1.1	–	0.86	0.15	0.86	0.47	1.19	5.64	–	–
SimPlan-2.0	0	0.1	0.16	0.25	0.54	0.06	0	–	–
LPG-1.2	0	0.28	0.06	0.14	0.17	0.05	0	–	–
Metric-FF	0	0.03	0.08	0.16	0.31	0.1	0	–	–
MIPS-3	0	0.13	0.05	0.22	5.01	0.35	–	–	–
SAPA-2	4.26	–	9.37	13.35	39.47	–	0.17	–	–
VHPOP-2.2	0.02	0.01	0.03	4.95	0.2	0.11	0.04	–	–
SGPlan-06	0	0.03	0.02	0.06	0.38	0.12	0	–	–
CPT-1.0	0.01	0.01	0.05	0.06	0.09	0.06	0	0.2	–
FastDown	0.07	1.59	0.34	2.48	3.3	1.87	0.06	0.08	–
LPG-TD	0.02	0.28	0.09	0.15	1.4	0.62	0.02	–	–
OPTOP	14.23	22.48	16.91	46.65	32.94	32.56	13.24	–	13.57
Satplan06	0.02	1.43	4.13	0.92	1.62	1.02	0.01	–	–

Somewhat surprisingly, two older planners (IPP4.1 and Prodigy) have the highest success rates for Hoffmann, Strict and Stek. On the other hand, these domains are all equally old and were largely available to the planner developers at the time. The older planners did not perform well on IPCs 3-5.

The time required to solve problems differs considerably between the old and more recent planners. Time out happened at 1800 seconds. The new planners are extremely fast, while the old planners are more likely to time out (see Table 6).

4.4.2. Do some of the planners offer distinct functionality?

As mentioned in the last section, 239 problems were solved by only a single planner. As Table 3 shows, only 12 planners solved problems that were solved by no other planner. Although the newer planners tend to solve more unique problems, some of the older planners (i.e., Prodigy and UCPOP) are still able to solve a few old problems that the more recent ones do not. FastDown and SGPlan excel on some of the IPC4 domains.

Fig. 2 presents a Sammon map across the planners. The planner vector was a 4726-dimensional bit-vector of planner success on each problem. Each planner is colored and shaped differently. The map illustrates how (dis)similar planners are to each other. For example, the partial order planners (POCL) are grouped in the upper left. The hybrid planners (HYBRID) tend to be near the planners of their conceptual lineage. The heuristic search planners (HS) are the most dispersed, which is most likely a result of the diverse heuristics and search algorithms represented by these planners.

4.4.3. Are some planners subsumed by others?

Some of the planners exhibit low success rates. A few of the planners were simply variations on each other (i.e., the next version or a different parameterization). We know that 12 planners solve problems not solved by any other; however, it seems likely that the functionality of a few (not in the set of 12) are completely subsumed by one of the more capable planners.

First, we pairwise compared the set of problems solved by the planners. Surprisingly, we find no cases where one planner's solved set was a subset of another's. Combinations of planners can cover the same problems of another planner, but no single planner was found to subsume any other.

Second, for those problems solved in common (that is, the union of solved problems) by each pair, we compare their times to determine whether one could be said to be significantly faster than the other. We use a one-sided matched-pairs Wilcoxon test with $\alpha = 0.05$ plus a Bonferroni adjustment of $\alpha/28$ to address the number of comparisons being made.[3]

[3] In this experiment, we followed the design of [6].

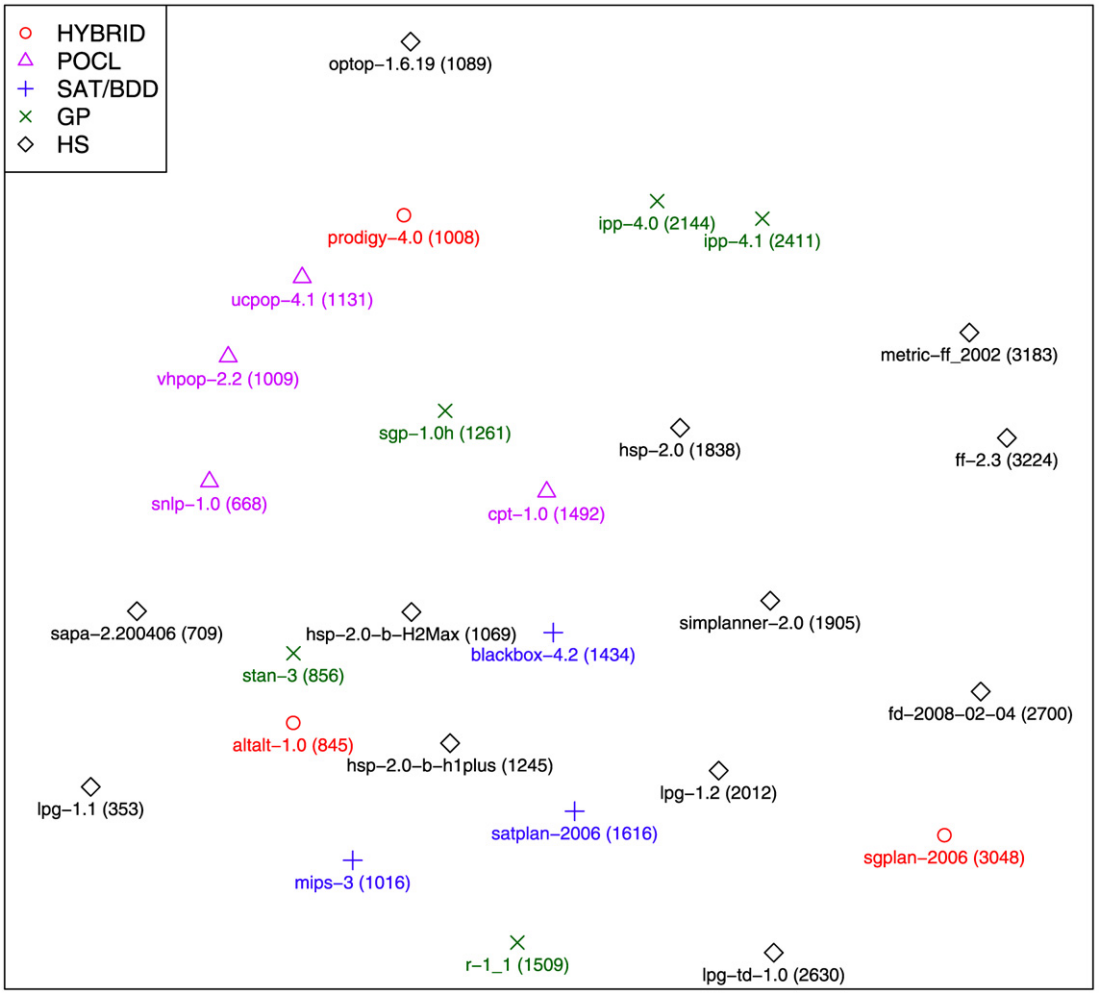


Fig. 2. Sammon map for domains with vector of number of problems solved by each planner. Planner points are colored and of a particular shape based on their primary search technology: POCL, SAT, GraphPlan (GP), Heuristic Search (HS), or a mixture (HYBRID). Prior to multidimensional scaling, each planner consisted of a 4726-dimensional bit-vector of whether the planner solved that problem.

We must also note that the comparison is restricted to only those problems solved in common, which means that in some cases that is a small number.

We find many cases in which one planner was faster than another. For example, SGPlan-06 is significantly faster than all of the other planners. FastDown and FF-2.3 are in the second level, and LPG-TD plus Metric-FF follow on the third level. The remaining planners are dominated by those just listed. They can also be ordered but they dominate over fewer of their successors. Notably, the speed seems divided along implementation language: the fastest planners tend to use C or C++ and the slowest planners tended to be those that used either LISP or Java. The relative ranking of a few of the planners showed that while they cannot solve a large percentage of the problems, what they do solve is done quickly.

4.5. Modeling classical planner performance

Our second goal is to learn planner performance models; these models offer insights into variability in planners and problems across the sets, predictability of the planners themselves, and which predictors correlate with difficulty.

4.5.1. Constructing the models

For each planner, we constructed two models: success and time. For success, we build a binary classifier (successful or not). Time predicts computation time needed for a given planner to attempt solving a given problem.

The problem/domain feature set began with features from [12] and [14]. However, we found the Howe et al. features to be insufficient for accurate models and the Hoffmann features, while powerful, to be intractable for larger problems. So we removed the Hoffmann features and added others based on our intuitions about what might influence performance.

Table 7
Planner performance and model accuracy data for success and time required using 10-fold cross-validation on the challenging problems (see the 'Challenge' columns) and the "train with old challenge problems, test on IPC5" (see the 'Test IPC5') columns. % fail and times are from the planner performance data; "Best % correct" and "Best RMSE" measure accuracy for the models selected as best.

Planner	Success models					Time models					
	Challenge			Test IPC5		Challenge				Test IPC5	
	% fail	Best % correct	Best classifier	% fail	Best % correct	Median time	μ time	Best RMSE	Best classifier	μ time	Best RMSE
AltAlt-1.0	99	99.51	NNge	100	100.00	0.00	0.64	2.77	KStar		
BlkBox-4.2	94	98.25	PART	92	80.09	0.00	21.64	124.20	KStar	0.43	307.07
CPT-1.0	99	99.58	JRip	100	100.00	0.13	14.02	90.71	KStar		
FastDown	51	94.12	RandFst	38	54.87	4.42	133.62	257.24	KStar	58.06	569.44
FF-2.3	58	95.31	J48	74	69.47	0.09	139.08	299.95	KStar	0.18	370.67
HSP-2.0	86	94.54	LMT	97	95.13	0.03	418.74	296.57	KStar	0.06	231.00
HSP-2.0r-h1plus	95	98.25	IB1	95	99.56	0.02	179.35	322.40	KStar	0.06	692.75
HSP-2.0r-h2max	98	99.02	KStar	100	96.90	0.02	335.19	388.57	KStar		
IPP-4.0	86	97.48	RandFst	98	91.15	0.16	412.52	369.39	KStar	0.16	609.88
IPP-4.1	85	97.69	RandFst	98	90.27	0.24	406.07	340.80	KStar	0.16	621.55
LPG-1.1	88	96.36	RandFst	92	92.48	0.01	211.65	224.83	KStar	0.13	174.31
LPG-1.2	85	96.22	KStar	91	91.15	0.02	199.15	246.97	KStar	0.14	198.57
LPG-TD	58	94.05	KStar	30	70.35	1.38	263.59	330.82	KStar	11.52	730.00
Metric-FF	60	94.12	KStar	73	58.85	0.18	56.40	183.03	KStar	0.20	278.32
MIPS-3	92	96.64	RandFst	96	96.02	0.08	366.67	306.10	KStar	0.56	557.32
OPTOP	94	97.13	IB1	93	93.36	15.52	63.91	71.81	KStar	34.19	237.74
PROD-4.0	99	99.72	DT	100	100.00	0.79	566.94	155.02	KStar		
SystemR	93	99.44	IB1	94	81.86	8.71	429.67	192.53	KStar	21.27	784.74
SAPA-2	99	99.23	RandFst	100	100.00	0.08	227.36	223.15	KStar		
Satplan06	86	95.38	KStar	76	76.11	0.02	137.15	335.90	KStar	33.55	1140.70
SGP-1.0b	98	99.02	Log	99	95.13	13.84	662.83	259.09	KStar	1798.95	1403.17
SGP-1.0h	98	99.02	Log	99	95.13	13.73	661.62	252.80	KStar	1798.94	1407.71
SGPlan-06	50	91.46	LMT	23	59.73	0.32	175.70	828.54	KStar	0.61	2037.41
SimPlan-2.0	86	92.65	LMT	96	96.02	0.19	40.69	129.52	KStar	0.15	164.27
SNLP-1.0	100	99.86	MLP	100	100.00	0.40	753.00	142.09	KStar		
STAN-4	99	99.30	PART	100	85.40	0.00	7.88	100.53	M5Rule		
UCPOP-4.1	98	99.09	NNge	100	100.00	2.26	709.95	218.95	KStar		
VHPOP-2.2	100	99.51	MLP	100	100.00	21.41	39.66	42.21	KStar		
Average	86.9	97.21		87.6	88.18	3.00	272.67	240.59		197.86	645.23

Each problem instance is defined by 32 features that can be automatically extracted from PDDL problem and domain definitions. In the description below, the name is followed by a number indicating how many features are based on it. The first three features are problem features, but the remaining are calculated from the domain description. The features are fast to extract, requiring only 0.0025 seconds on average.

goals (1): A count of the number of goals to achieve.
objects (1): A count of the number of objects in the problem instance.
inits (1): A count of the number of items in the starting state of the problem instance.
operators (1): A count of the number of operators in the domain.
predicates (1): A count of the number of predicates in the domain.
predicate arity (3): The (min,max,average) of the number of parameters over all predicates.
predicates in precondition (3): The (min,max,average) of the number of predicates in the precondition over all operators.
predicates in effects (3): The (min,max,average) of the number of predicates in the effects over all operators.
negative effects in postcondition (3): The (min,max,average) of the number of negative predicates in the postcondition over all operators.
operators with negative effects (2): The (count and ratio) of (unground) operators with negative effects.
requirements (13): Whether the domain required specific features of PDDL: `ADL`, `conditional-effects`, `derived-predicates`, `disjunctive-preconditions`, `domain-axioms`, `equality`, `existential-preconditions`, `fluents`, `quantified-preconditions`, `safety-constraints`, `STRIPS`, `typing`, `universal-preconditions`. As noted in [12], some domains fail to state clearly these requirements while others state a requirement that is never used. We corrected for `typing` requirements if the domain actually used types but did not require it.

Using the entire problem set for learning was going to bias the models towards the more common and frankly less interesting simple problems. Consequently, we restrict our learning set to just Challenge problems. We tried 32 different models from WEKA [15]; WEKA is a data mining toolkit written in Java that provides a standard format for running common machine learning algorithms. Table 7 highlights the models that were most accurate for the test used to select models: ADTree (ADT), Conjunctive Rules (ConjRul), Decision Table (DT), Decision Stumps (DecStmp), GaussianProcess (GP), IB1, J48, JRip, KStar, LMT, Logistic Regression (Log), Rules from Model Trees (M5Rule), MultilayerPerceptron (MLP), OneR, NNge, and PART, Random Forests (RandFst), RBF Networks (RBFNt), Ridor, Simple Logistic Regression (SimLg), and SMO. They include lazy

instance learners, rule based, decision tree, model tree, nearest neighbor, kernel, neural network and regression algorithms. WEKA is fast to learn and run the learned models. The mean time for WEKA to learn the models is about 4.3 seconds; prediction of a single data point is usually a millisecond or less.

4.5.2. 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.

To evaluate the models, we performed 10-fold cross-validation on the Challenge data. We also evaluated whether the models generalize to the newest competition by training on all but IPC5 and testing on the IPC5 problems. Table 7 shows the results of the full Challenge experiments ("Challenge" columns) and the IPC5 holdout experiment ("Test IPC5" columns).

The models of success are quite accurate. Guessing "fail" can be viewed as a baseline (based on the 'prior') model – it is the most likely outcome for all the planners – and then % fail is its % correct. The best classifier's % correct is higher than (or within 1%) of the prior for all models.

In some cases, the % correct is considerably higher than the prior: FastDown, FF-2.3, LPG-TD, Metric-FF and SGPlan-06. A two-tailed paired sample t-test comparing the correctness of the baseline against the best model shows the baseline's accuracy is significantly lower ($t \approx -3.89$, $p < 0.001$). Moreover, the average correctness is quite high: 97.21% over all planners. The best classifier varies across the planner set, but picking the best classifier does matter as the average accuracy across all classifiers and planners was slightly lower: 94.33%.

The models of time are not so accurate. The best Root Mean Squared Error (RMSE) averages more than the average time. The highest RMSE is 828.54 seconds (SGPlan-06), which is half of the highest possible time of 30 minutes. The next lowest RMSE is 388.57. However, not all planners are so hard to predict. The lowest RMSE is 2.77, and 12 models have RMSEs lower than the average times. 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.

4.5.3. Do the models generalize to new problems? (Train on Old, Test on New)

We trained models on the portion of the Challenge data other than the latest (226 problems from the IPC5 subset) and tested on these newer problems. As Table 7 shows, the planner failure rates diverge on the newer problems; the difference in average accuracy suffered from a gap of $97.21 - 86.92 = 10.29$ to a gap of $88.18 - 87.64 = 0.54$. Additionally, 14 success models had accuracy within 1% of the prior and only 5 models exceeded the prior. This result suggests that the problems are substantially different.

Many of the planners are unable to solve problems from the IPC5 set (see the '% Fail' column within the 'IPC5 Test' column of Table 7). We do not calculate time models for these planners. For the planners that succeeded at least once the time models were much worse on the test set except for two planners (SGP-1.0b and SGP-1.0h). Thus, generally the predictions are dissimilar, which provides more evidence that the problems from IPC5 are indeed different and the models are not generalizing.

5. Linking planner performance to search topology

Previous sections have offered hints toward explanations of performance. In this section, we demonstrate a proof-of-concept that these performance data can be used to empirically validate models of search topology in Classical Planning. Hoffmann analyzed *whether* the success of Fast Forward (FF) was explained by local search topology. He provided both empirical [14] and theoretical [16] analyses showing that many common benchmarks were easily solvable (some even linearly) by FF because either they lack significant minima or the heuristic easily addresses them [8]. Recently, Hoffmann extended these results to the newer problems from IPC3 and IPC4 [17]. Hoffmann's analysis was convincing in its findings for FF. There is good reason to believe it might extend to other planners. We ask the question: Is the performance of heuristic search planners (as a group) sensitive to the taxonomy axes?

We apply Hoffmann's two-dimensional taxonomy to 11 planners on the 30 domains. Table 1 ('h^+' column) marks with an asterisk the 11 heuristic search planners that directly use an approximation of the h^+ heuristic. Table 8 (right-most two columns) reproduces Hoffmann's taxonomy as a flat table for the 30 domains he studied, while Table 9 shows a summary of the problems in each category. 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 HU:MT pairing are among the most challenging while those in the lowest pairing HC:MB are among the most simple. We focus our effort on the Challenge problems which causes several domains to be dropped from the table.

5.1. Empirical analyses

To answer our question of sensitivity to the taxonomy, we statistically measure the effect of both topological axes in terms of ratio of success and running time. We judge a test significant if $p < 0.05$ and highly significant if $p \ll 0.001$.

Table 8
A flattened representation of the domains and problems we examine. The rightmost columns show the placement of each domain into the appropriate taxonomy category, ordered first by dType {HC,HH,HR,HU} and then by mType {MB,ML,MT}. Within each taxonomy pair, the domains are ordered in approximately increasing difficulty. The second and third columns list number of problems each domain contributes to the problem sets.

Domain (Symbol)	Problem set *mO1*	Labels	
		dType	mType
Gripper	–	HC	MB
Ferry	–	HC	MB
Logistic	18	HC	MB
Blks-3op	–	HC	ML
Fridge	1	HC	ML
Briefcase	–	HC	ML
Hanoi	–	HC	ML
Blks-4op	66	HC	MT
Depot	29	HC	MT
Driverlog	15	HC	MT
Mic-SIM	–	HH	MB
Mic-Str	–	HH	MB
Movie	1	HH	MB
Tyreworld	1	HH	MB
Satellite	45	HH	MB
TSP	–	HH	MB
Zeno	54	HH	MB
Grid	4	HH	ML
PSR	157	HH	MT
Pipesworld	129	HH	MT
Phil	127	HR	MB
Schedule	126	HR	MB
Protocol	11	HR	MT
Rover	21	HR	MT
Airport	53	HU	MT
Assembly	135	HU	MT
Freecell	60	HU	MT
Mic-ADL	97	HU	MT
M-prime	20	HU	MT
Mystery	15	HU	MT
TOTAL	1185		

Table 9
Counts of the Challenge problems within each taxonomy category. Dashes indicate an impossible category of this taxonomy.

mType	dType				Total
	HC	HH	HR	HU	
MT	110	286	32	380	808
ML	1	4	0	–	5
MB	18	101	253	–	372
Total	129	391	285	380	1185

We group all problems attempted by the 11 h^+ planners according to the dType and mType then perform our statistical analysis. To test for an effect on success ratio (the dependent variable), we produce a contingency table of successes and failures grouped by the category of interest (independent variable) then perform a G-test. A significant G-test can be interpreted that the taxonomy category effectively predicts success. Table 10 (second row) shows the results of the G-test followed by the contingency table that we constructed for the Challenge data.

We also examine the runtime for these problem instances by grouping the data by each type and performing several tests. The results for runtime analysis of the h^+ planners on Challenge are summarized in Table 11. We start with a one-way analysis-of-variance (ANOVA) test on a linear model of the data (this is a common and fairly robust assumption even with deviations from Normality or the presence of unequal variances). For this test, we report the F-statistic (ANOVA's statistic) on the time-to-completion (TTC), time-to-success (TTS), and time-to-failure (TTF). The top three rows of Table 11 show the results of these ANOVA tests. A significant ANOVA implies a true difference over *all* groupings and justifies a pair-wise comparison.

For pair-wise comparisons, we use the TukeyHSD test because it is conservative in assigning grouping similarities; this test is reported as p-value adjusted to control the experiment-wise error at $\alpha = 0.05$. An insignificant TukeyHSD pair-wise

Table 10
G-test and Contingency Table of h^+ planners for Challenge data.

mType				dType		
$G = 148.81,\ p \ll 0.001$			A	$G = 23.04,\ p \ll 0.001$		
	Succeed	Fail			Succeed	Fail
MB	2963	1129		HC	861	558
ML	25	30		HH	2950	1351
MT	6692	2196		HR	2499	636
				HU	3370	810

Table 11
ANOVA and Pairwise comparisons on h^+ planners for the Challenge data.

mType					dType				
$F = 46.33,\ p \ll 0.001$				TTC	$F = 134.14,\ p \ll 0.001$				
$F = 9.69,\ p \ll 0.001$				TTS	$F = 7.52,\ p \ll 0.001$				
$F = 38.35,\ p \ll 0.001$				TTF	$F = 196.74,\ p \ll 0.001$				
	MB	ML	MT			HC	HH	HR	HU
MB	–	cs			HC	–	s		s
ML	cf	–	cs		HH		–		s
MT		cf	–		HR			–	c
					HU			cf	–

comparison indicates that the two groupings are significantly *similar*. Table 11 (bottom portion) shows a summary of the pairwise comparisons of the groups for TTS ('s') in the upper triangle of each sub-figure, TTF ('f') in the lower triangle, and TTC ('c') in both. A letter for a pairing indicates that TukeyHSD **did not** find a significant difference between them – it means they are statistically similar.

The TukeyHSD results identify a gap between ML and MT for all three categories of time to success, failure and completion. This validates the distinction in the taxonomy between plateaus of bounded length (ML) and plateaus that are local minima (MT). The results for dType are more mixed where the success runs show similarity across the extremes. Also, the distinction between HR and HU is less supported by these data.

Our data suggest that Hoffmann's taxonomy predicts the performance of other heuristic search planners using approximations of the h^+ heuristic. Although all the h^+ planners use a similar heuristic, each planner applies the heuristic and performs search in a distinct way. It may be possible to account for more variance by grouping the planners according to subfamilies based on search algorithm or other planner features.

A number of limitations cause us to view our results as suggestive rather than definitive; future work will focus on alleviating these limitations. It is clear that the planning systems are complex and not designed with large-scale comparisons in mind. Further, intra-domain problem difficulty is hard to assess, and there may be an effect due to the existence of simple (or challenging) problems in one or more domains (for example, from a single grouping) that lead to success or failure. Low cell counts and missing/sparse data limit our inferences, though it is noteworthy that we were able to show a significant effect with the data we already obtained.

6. An analysis of some newer problems

Having discovered in our analyses that too few of the current problems adequately differentiate performance by time, we take up the challenge of constructing some new problems. We roughly follow the methodology set out by Taillard in producing the well known difficult Job Shop scheduling problems in the OR Library [18]: generate problems by varying parameters, test the resulting problems on some existing solver, and retain only those that are hard to solve.

First, we select four domains with problem generators: *briefcase*, *ferry*, *schedule*, and *tyreworld*. These domains span the difficulty level of Hoffmann's taxonomy. For each of these, we used the FF Problem generators to construct problems by varying the parameters as follows: briefcase, number of locations and portables [50, 69]; ferry, the number of locations and cars [50, 69]; schedule, number of objects [40, 59]; tyreworld, the number of hubs, nuts, and boots [50, 69].

Second, we construct a new domain. We started with a relatively simple domain and problem description for the SIPE planner [19] (an HTN planner with a considerably more expressive language than PDDL) and translated it into PDDL by making a lot of simplifying assumptions. The **sipe-travel**[4] domain consists of moving passengers around a map of cities using a variety of transports. Each passenger may have one or more destination goals, at which the passenger can book accommodations. In the problems, we varied the number of passengers in the domain in the range [40, 59], but gave every

[4] We thank David Wilkins of SRI International for the SIPE Logistics Domains. We also thank Christie Williams who primarily wrote the SIPE translation and worked with the problem generators to create this problem set.

Table 12
Success statistics for each of the domains and the whole set. The columns show the total number of problems in each set (*numProblems*), the number of problems solved by half the planners (*solvedByHalf*), the ratio of success over all (*Suc*), the ratio of success within one second (*1Sec*) and ten seconds (*10Sec*), and the number of planners solving at least one problem in the set (*Plan+1*).

Domain	numProbs	solvedByHalf	Suc	1Sec	10Sec	Plan+1
hard	100	0	0.11	0	0.03	8
briefcase	20	0	0.05	0	0	3
ferry	20	0	0.26	0.02	0.11	8
schedule	20	0	0.07	0	0.04	2
sipe-travel	20	0	0.07	0	0	2
tyreworld	20	0	0.08	0	0	3

Table 13
Time statistics for each of the five domains taken together and separated by success and failure.

Domain	All			Successes			Failures		
	mean	sd	med	mean	sd	med	mean	sd	med
hard	301.2	607.8	4.2	135.2	299.8	22.2	321.0	631.8	1.3
briefcase	266.0	580.1	0.8	700.7	489.3	636.4	243.9	576.1	0.4
ferry	363.0	632.5	24.1	95.5	257.1	14.7	459.1	696.7	68.3
schedule	98.4	396.6	0.1	9.9	6.6	7.5	105.3	410.8	0.1
sipe-travel	459.1	710.4	24.9	119.0	151.3	26.2	485.6	729.8	24.1
tyreworld	321.3	617.9	10.8	48.3	26.3	46.2	344.0	637.8	6.1

passenger the same goal of starting at Honolulu, visiting Seattle, London, Palo Alto, and ending at Los Angeles. We model neither capacity constraints nor edge costs in this version of the problem; all resources have infinite capacity, and all travel costs are unit.

We identified eight representative planners by finding a set that together solved all the problems in the challenge set: FF-2.3, HSP-2.0, HSP-2.0r-h1plus, LPG-TD, Metric-FF, MIPS-3, SystemR, and SGPlan-06. Then we selected problem instances[5] if one or more of the eight representative planners solved the problem in no less than 10 seconds but not more than 30 minutes.

For these five domains, we examine a subset of the previous questions we asked of the performance on the benchmark problems. Performance results were collected for these new problems in the same manner as with the benchmark problems, but only on the eight planners.

6.1. Does the performance on the new "hard" problems differ from previous problem instances?

Tables 12 and 13 show the overall results for the five domains. The overall planner success ratio has degraded considerably when compared to the benchmark results (as in Tables 16 and 17). Additionally, the times required are considerably higher than before; in fact, they are considerably higher than times for any previous domains.

Some domains are solved only by two planners. For all but five planner/domain combinations, planners either solve all or none of the problems in a domain. Even for the fairly simple Ferry, the number of problems that half the planners solved went to zero from all problems in the original set being solvable by half the planners. The one exception is schedule, which shows only a slight difference in solvability and time. Generally, then, we succeeded in creating harder to solve problems.

6.2. Is sipe-travel significantly different than others?

The SIPE travel domain shares a common ancestor in the so-called "logistics" domains, wherein the objective is to move objects (such as people, packages, or vehicles) among locations (such as depots, ports, or cities). Given this similarity along with the explicit parallelism of the non-interacting goals in the domain, it is somewhat surprising that the planners do not perform well on sipe-travel problems. It presents a novel, but tractable, challenge for LPG-TD where the median runtime was considerably higher than the other two logistics domains; recall, however, that LPG-TD is a stochastic search planner and may have done better in a set of cumulative runs. But SGPlan-2006 solves the domain instances relatively quickly. The latter result suggests that sipe-travel is amenable to a planner that uses a divide-and-conquer paradigm.

6.3. Do some of the planners offer distinct functionality?

The results in Tables 14 and 15 start to highlight performance differences that can be explained by planner technologies and implementations. For example, HSP-2.0r-h1plus solves the Ferry domain faster and more successfully than HSP-2.0r-

[5] The problems are available on our website: http://www.cs.colostate.edu/meps/repository.

Table 14
Planner success ratio for each of the five domains.

Planner	BRIEFCASE	FERRY	SCHEDULE	SIPE-TRAVEL	TYREWORLD
FF-2.3	1	1	1	0	1
HSP-2.0	0.03	0.47	0	0	0
HSP-2.0r-h1plus	0	0.7	0	0	0
LPG-TD	0	1	0	1	0.15
Metric-FF	0	1	1	0	0
MIPS-3	0	1	0	0	0
SystemR	0	1	0	0	0
SGPlan-06	0.25	1	0	1	1

Table 15
Planner median time to success for each of the domains. A dash indicates zero successes for that planner on that domain.

Planner	BRIEFCASE	FERRY	SCHEDULE	SIPE-TRAVEL	TYREWORLD
FF-2.3	681.12	4.08	5.52	–	25.7
HSP-2.0	1453.63	221.57	–	–	–
HSP-2.0r-h1plus	–	106.05	–	–	–
LPG-TD	–	11.73	–	206.14	51.62
Metric-FF	–	9.23	10.07	–	–
MIPS-3	–	27.59	–	–	–
SystemR	–	36.59	–	–	–
SGPlan-06	159.86	0.84	–	17.91	65.9

h2max; for Ferry, regression search with an admissible heuristic may be the best choice. For the other domains, HSPr is not successful. The relative ranking of the planners based on the median runtime changes depending on the domain; for example, consider in Table 15 the rankings of FF-2.3, LPG-TD, and SGPlan-06 on Ferry and Tyreworld. So it is not always the case the best planner for one domain is best for another.

The failure times shown in Table 13 reveal still low median runtimes for most of the planners. Many planners are likely running out of memory early in the search.[6] This suggests that increasing the problem size actually may work against certain planning technologies simply because of memory requirements. None of these planners use a lifted representation, which is often used to handle memory explosion.

Differences in implementation details can lead to a dramatic performance difference: LPG-1.1 and LPG-1.2 are absent from the list, but the newer LPG-TD is present. In contrast, FF-2.3 performs very well for the four extended domains; yet, Metric-FF, which uses an internal engine modified from FF, has either memory or time requirements such that it fails to perform well. Many planners perform poorly on the new problem set. Even planners with relatively subtle implementation differences can exhibit dramatic performance differences.

7. Future work

We have a great deal more we can do with the data that would support the theory and implementation of planners. We present key points of continuing work that we hope will identify dependencies between the domains, heuristics, algorithms, and runtime dynamics in classical planning.

7.1. Planners

As mentioned in the planning section, our complete list of planners is much longer than the actual list we have run so far; we have 86 versions, of which about 40 remain to be tested, and this does not include all of the planners from the two most recent competitions. Our hope is to continue adding planners as resources and time permit.

We recognize that our efforts so far have focused on satisficing planners. Indeed some of the planners in our list are optimal planners. A clear extension of these results would be to further distinguish performance by this criteria.

7.2. Problems

A key issue is to expand the problem set in ways that are both challenging and realistic (for clarification on this point, see [20]). Some promising steps in this direction are domains for high speed printing [21], for workflow management of a job site [22] and network intrusion [23]. We have downloaded these problems and intend to explore them alongside

[6] Memory is clearly important for many planners. Unfortunately, it is hard to gauge and control for memory in planner performance because of the cache, OS version, and the compilers. Thus, we did not explore that issue in our studies.

the current benchmarks. We have also obtained the problem generators for many of the IPC domains as well as a random problem generator from Sungwook Yoon.[7]

Some studies examine particular qualities of domains that impacted planner performance. One study along these lines was examining linkability of actions in a comparison of total-order and partial-order planners [24]. We hope to identify these kinds of domain specific qualities. Generating specific problems with IPC problem generators will be useful for testing specific hypotheses about the dependencies we notice in existing benchmarks.

Another valuable approach is to identify specific characteristics shared by some domains (such as resource constraints in logistics or time in schedule) and generate problems by manipulating them. This has the advantage of being more hypothesis driven and focused.[8]

7.3. Models

Given the relatively high accuracy for the success models but low for the time, we need to examine the features to determine how they are influencing the models. A key issue is likely to be the preponderance of domain versus problem features. We expect to more closely examine the learned models to determine which features are most important in each of the models and try to explain why we see the accuracy differences.[9]

Given the relatively poor performance of the time models, we need to look more closely into meta-learning or feature selection techniques in machine learning. The current features may be overly simple for the time models, and another approach is to expand the feature set. We have started to examine two kinds of features that we believe may improve the runtime models considerably. The first are features of planner capabilities and implementation details. Second, we have recently discovered we might be able to approximate the memory footprint of the final (relaxed) planning graph to help gauge its contribution to performance.

Beyond these simple feature extensions, a reasonable place to start extending the feature set may be including some features from HAP [26], TIM [27], Londex [28] and LPG [29]. A more sophisticated approach would be to leverage structural features from the causal graph.[10]

7.4. Relaxing classical assumptions

Because of the large number of planners and problems available, these studies are based in the classical planning paradigm; numerous extensions to PDDL relax classical assumptions [5,6,11]. There is also the separate probabilistic track of the competitions. The number of planners implementing these features is (slowly) growing, so there is hope that we could eventually extend the kinds of analyses beyond classical planners.

7.5. Extending the analyses

The recent IPCs have started a 'tradition' of thoroughly analyzing the results. Similarly, researchers have analyzed specific planners; such as FF [8], TP4 [30], FastDown [31], Marvin [32], etc. Complementary to these efforts, our goal in this survey was to assess the field of Classical Planning from a wide perspective and to understand trends related to comparative evaluation. As we did with Hoffman's topology, we hope to examine specific hypotheses about what makes planners work well through generalizing previous results on single planners to planners with similar mechanisms.

8. Summary

In this paper we have described a large retrospective and prospective study of the state of the art in classical planning (STRIPS and ADL). We chose classical planning because it is where most research has been done in planning and the technologies developed often underlie extensions as well.

The IPC organizers and planner developers are to be commended for making available extraordinary resources for supporting comparative evaluation and analyses of performance. However, aside from a handful of notable exceptions – such as those found in post-hoc analyses of the competitions [5,6], topological analyses of state space planning [17], and complexity analyses of the planning domains [33,34] – few comprehensive assessments, analyses and models have been done. The scope of the compendium (its life span, size and varied, sometimes ad hoc roots) work against careful experimentation.

Our contribution is to characterize the compendium and suggest how data from a large study can be used to inform subsequent experimentation, and validate explanatory models. Specifically, we offer the following observations from our studies.

[7] Personal communication.

[8] We thank "Reviewer 1" for this suggestion.

[9] A recent paper *What Makes Planners Predictable*? examines the models in more detail [25].

[10] We thank David Smith for this suggestion.

- The chronological comparisons (difficulty of old versus new problems in the benchmarks and performance of old versus new planners) have confirmed that the field has made significant progress in terms of the difficulty of problems that have been proposed and the capabilities of planners.
- Statistical models, such as summary statistics on problem difficulty relative to different planners and Sammon maps for visualizing how differentiated are problems and planners, can help guide researchers in principled design of comparative experiments.
  - Any problem set should include challenging problems that are likely to differentiate performance and/or that are suited to the design goals of the planner innovation.
  - Domains should be selected carefully to highlight specific capabilities.
  - Although no planner always dominated in our study, many are extremely limited in what they can solve. To support fair comparisons, researchers should select planners (sometimes more than one) for comparison that appear to perform well on the domains/problems in the target set.
- Inexpensive problem/domain features coupled with automated learning produce very accurate predictive models of success for planners.
- Inexpensive problem/domain features coupled with automated learning produce weakly predictive models of time to success for planners. Clearly, the mechanisms underlying time are more complex.
- The field needs new challenging problems, which can be constructed from existing domains or by translating from other planner paradigms.
- Our analysis of the performance data verifies that Hoffman's topology partially explains the performance of heuristic search planners that use approximations of the h^+ heuristic.

Together these observations raise intriguing questions about how problems can and should direct our research. While useful for identifying suitable planners for comparison, the predictability of current performance is also disturbing when considered as an indicator of how hard we are pushing the envelope. IPC organizers have shouldered much of the burden of developing new problems.

The community needs to reconsider how we relate problem performance to problem structure: both in terms of understanding and explaining performance now and extending beyond the current domain requirements and metrics. Steps in the right direction have been taken in recent comprehensive analyses of the competitions that incorporate alternative metrics. Unfortunately, many planners disregard the temporal or quality information during search. Planning and the areas of AI that use planning will benefit enormously by our broadening the kinds of problems that we can solve and changing how we compare planners.

Acknowledgements

We wish to thank Landon Flom, a (now graduated) Masters student at CSU, who wrote the scripts to run and analyze the WEKA models. We thank the anonymous reviewers for this special issue as well as several recent AAAI and ICAPS workshops in which preliminary versions of this work appeared. We also thank the ICAPS community – notably Maria Fox, Jeremy Frank, Jörg Hoffmann, and David Smith – for conversations and suggestions that helped guide and shape this work. Mark Roberts would like to thank the AAAI Doctoral Consortium and the mentors – especially Marie des Jardins – for excellent feedback on portions of this work. Most importantly, we thank the authors of the planning systems and the organizers of the competitions, without whom this work would be impossible.

Both authors were supported for portions of this work by grant number IIS-0138690 from the National Science Foundation and grant number FA9550-07-1-0403 from the Air Force Office of Scientific Research. Adele Howe was also supported by grant number FA9550-08-1-0422 from the Air Force Office of Scientific Research.

Appendix A. Performance data appendix

Table 16
Success statistics across all problems and all planners. The columns show the total number of problems in each set (*numProblems*), the number of problems solved by half the planners (*Half*), the ratio of success over all (*Suc*), the ratio of success within ten seconds (*10Sec*) and one second (*1Sec*), and the number of planners solving at least one problem in the set (*Plan+1*).

Domain	Count	Half	Suc	10Sec	1Sec	Plan+1
IPC1	155	50	0.4	0.33	0.26	27
grid	5	1	0.39	**0.30**	**0.17**	18
gripper	20	5	0.45	0.44	0.43	24
logistics	35	9	0.39	**0.31**	**0.22**	22
movie	30	0	0.24	0.22	0.22	7
mystery-prime	35	20	**0.51**	**0.40**	**0.27**	25
mystery	30	15	0.40	**0.32**	**0.24**	26

(*continued on next page*)

Table 16 (continued)

Domain	Count	Half	Suc	10Sec	1Sec	Plan+1
IPC2	645	88	0.25	0.21	0.18	26
blocks	103	27	0.32	**0.24**	**0.19**	24
freecell	60	5	0.21	**0.13**	**0.09**	14
logistics	32	18	0.48	0.46	0.44	17
miconic	300	38	0.27	0.23	**0.21**	21
schedule	150	0	0.11	0.10	0.08	9
IPC3	440	84	0.29	0.22	0.17	28
depot	88	16	0.21	0.16	**0.12**	25
driverlog	80	23	0.27	**0.22**	**0.18**	27
freecell	40	4	0.27	**0.17**	**0.10**	20
rover	80	6	0.37	**0.32**	**0.25**	19
satellite	72	9	0.21	0.17	**0.14**	22
zeno-travel	80	26	0.40	**0.27**	**0.20**	27
IPC4	858	52	0.12	0.09	0.06	25
airport	50	0	0.08	0.05	**0.03**	5
airport-fixed	50	3	0.16	0.12	**0.09**	16
civ	20	0	0.08	0.05	**0.03**	3
grd-simple-adl-psr	50	0	0	0	0	1
grd-protocol	110	0	0.09	0.06	**0.04**	8
grd-psr	100	21	0.26	0.24	**0.19**	19
pipesworld	100	12	0.25	**0.16**	**0.11**	21
protocol	192	0	0.03	0.02	0.01	5
psr	150	0	0.03	0.02	0	5
satellite	36	16	0.42	**0.32**	**0.24**	23
IPC5	450	22	0.17	0.13	0.10	27
grd-pathways-prop	30	4	0.15	0.11	**0.09**	15
grd-openstacks	30	0	0.11	0.10	0.08	12
grd-pipesworld	50	1	0.06	0.03	0.02	14
grd-rover	40	3	0.19	**0.14**	**0.11**	17
grd-tpp-prop	30	5	0.23	**0.18**	**0.16**	23
grd-trucks	30	0	0.05	0.02	0.01	6
openstacks	30	0	0.20	0.16	**0.12**	8
pathways-prop	30	0	0.14	0.13	0.10	9
pipesworld	50	0	0.16	**0.09**	**0.06**	16
rover	40	3	0.34	**0.27**	**0.20**	20
storage-prop	30	6	0.28	0.24	**0.20**	16
tpp-prop	30	0	0.24	**0.17**	**0.13**	11
trucks	30	0	0.07	0.05	0.04	8
Hoffmann	1814	967	0.53	0.5	0.48	27
assembly	135	0	0.07	0.07	0.07	2
bw-3ops	150	**150**	**0.86**	0.85	0.85	26
bw-4ops	150	**150**	**0.84**	0.82	0.80	27
briefcase	115	0	0.36	0.34	0.33	13
ferry	120	**120**	**0.84**	0.83	0.82	25
freecell	110	**110**	**0.64**	**0.59**	**0.50**	20
fridge	12	0	0.32	0.28	0.28	9
grid	139	**139**	**0.83**	0.80	**0.73**	26
gripper	7	**7**	**0.77**	0.74	**0.72**	24
hanoi	5	3	**0.59**	**0.51**	**0.51**	26
logistics	101	**101**	**0.79**	0.77	**0.70**	25
miconic-adl	107	0	0.18	0.15	0.15	8
miconic-simple	113	0	0.25	0.21	0.21	7
miconic	101	15	0.46	0.42	0.42	20
movie	30	**30**	**0.65**	0.62	0.62	19
mprime	107	45	0.41	**0.36**	**0.35**	17
mystery	112	89	**0.54**	**0.49**	**0.47**	17
schedule	191	0	0.25	0.23	0.22	10
tsp	8	**8**	**0.74**	**0.69**	**0.67**	21
tyreworld	1	0	0.46	0.46	0.46	13
Strict	327	174	0.35	0.34	0.33	27
art	12	**12**	**0.73**	0.69	0.69	21
art-1d	20	16	0.49	0.49	0.49	14
art-1d-rd	20	**20**	**0.54**	0.54	0.54	15
art-md	20	**20**	**0.50**	0.50	0.50	14
art-md-ns	20	**20**	**0.54**	0.54	0.54	15
art-md-ns-rd	20	**20**	**0.55**	0.54	0.54	16

Table 16 (continued)

Domain	Count	Half	Suc	10Sec	1Sec	Plan+1
art-md-rd	20	**20**	**0.54**	0.54	0.53	16
att-bw	2	0	0.38	0.34	**0.29**	11
att-bw2	1	**1**	**0.64**	0.64	**0.57**	18
att-logistics	6	4	**0.58**	**0.51**	**0.49**	21
bw	3	0	0.36	0.32	0.32	10
briefcase	3	**3**	**0.68**	0.64	0.64	19
briefcase-world	4	0	0.14	0.14	0.14	4
bulldozer	9	**9**	**0.54**	0.54	0.52	17
eight-puzzle	2	0	0.04	0.04	0.04	1
ferry	2	**2**	**0.61**	0.57	0.57	17
ferry-typed	1	**1**	**0.64**	0.61	0.61	18
flat-tire-adl	1	0	0.11	0.11	0.11	3
flat-tire	4	0	0.26	**0.21**	**0.21**	8
flat-tire-typing	6	0	0.16	0.15	0.15	11
fridge	2	0	0.04	0.04	0.04	2
fridge	2	0	0.41	0.38	**0.36**	12
homeowner	1	0	0.07	0.07	0.07	2
logistics	6	**6**	**0.67**	**0.62**	**0.60**	21
mcd-bw-axiom	2	0	0	0	0	0
meet-pass	5	2	0.35	0.34	0.33	19
molgen-adl	1	0	0.04	0.04	0.04	1
molgen	2	0	0.14	0.14	0.14	4
monkey	3	2	0.40	0.37	0.36	18
montlake	1	0	0.07	0.04	0.04	2
morris	2	0	0.23	0.23	0.23	13
movie	1	0	0.07	0.07	0.07	2
mystery	1	0	0	0	0	0
mystery	41	0	0.05	0.04	0.03	17
mystery-typed	41	0	0.05	0.03	0.02	15
people	1	0	0.18	0.18	0.18	5
prodigy-bw	4	2	0.46	0.46	0.43	15
road-operators	1	**1**	**0.93**	0.89	0.89	26
roads	2	1	0.29	0.27	0.27	16
safety-test1	1	0	0.14	0.11	0.11	4
safety-test2	1	0	0.14	0.11	0.11	4
simple-blocks	4	**4**	**0.57**	0.54	0.54	16
simple	7	0	0.03	0.03	0.02	2
snlp-bw1	4	**4**	**0.60**	0.56	0.56	17
snlp-bw2	4	**4**	**0.54**	0.54	0.52	16
trains	3	0	0.21	0.21	0.20	6
t-trains	3	0	0.18	0.18	0.17	7
uni-bw	4	0	0.11	0.11	0.11	3
woodshop	1	0	0.14	0.14	0.14	4
Sodor	13	0	0.17	0.14	0.12	13
three-way	13	0	0.17	0.14	0.12	13
STEK	24	0	0.04	0.03	0.01	4
stek-all-action	24	0	0.04	0.03	0.01	4

Table 17
Time statistics across all planners and all problems.

Domain	All			Successes			Failures		
	mean	sd	med	mean	sd	med	mean	sd	med
IPC1	344.2	677.2	0.98	30.6	142.9	0.25	549.2	799.1	13.3
grid	541.7	790.1	19.46	79.8	296.7	1.3	831.7	863.3	98.22
gripper	478.4	764.6	0.3	7.3	87.9	0.03	863.9	852.6	432.92
logistics	492.0	777.6	9.3	26.2	107.1	0.57	784.5	869.3	46.8
movie	31.3	223.3	0	21.8	137.4	0.1	34.3	243.8	0
mystery-prime	335.0	652.2	3.27	45.5	151.2	0.69	632.9	815.5	48.7
mystery	373.2	695.5	3.17	28.3	160.6	0.38	603.2	810.7	30.75
IPC2	266.1	599.3	0.13	25.1	120.4	0.14	344.9	668.2	0.12
blocks	498.4	756.4	16.88	46.1	175.4	0.31	707.2	827.1	61.82
freecell	664.4	789.5	68.57	78.0	219.2	1.88	821.5	812.9	648.27
logistics	151.3	487.9	0.06	5.5	44.3	0.03	284.5	645.9	0.12

(*continued on next page*)

Table 17 (continued)

Domain	All			Successes			Failures		
	mean	sd	med	mean	sd	med	mean	sd	med
miconic	166.0	474.5	0.06	14.3	75.2	0.11	223.2	544.0	0.04
schedule	172.1	518.7	0.06	15.1	90.7	0.15	191.1	545.3	0.06
IPC3	448.2	737.0	4.63	42.6	161.6	0.4	613.8	812.2	19.44
depot	507.1	764.6	13.79	50.7	198.0	0.52	630.4	812.7	32.16
driverlog	481.7	748.3	10.13	47.7	196.5	0.15	643.4	811.1	41.27
freecell	691.4	807.6	76.78	63.8	159.0	2.62	923.2	827.4	1087.38
rover	447.7	751.2	1	24.1	134.9	0.24	691.4	846.1	19.74
satellite	204.6	549.3	0.02	26.5	108.4	0.16	250.6	605.3	0
zeno-travel	448.1	732.2	11.7	51.5	153.5	1.13	716.9	839.5	43.83
IPC4	147.2	457.9	0.08	42.7	148.2	0.81	160.8	482.3	0.05
airport	100.8	383.5	0.06	74.5	167.2	1.96	103.1	396.7	0.04
airport-fixed	206.4	520.2	0.32	33.1	105.9	0.71	240.2	560.7	0.24
civ	26.7	162.4	0	53.6	192.9	2.42	24.5	159.6	0
grd-adl-psr	4.1	23.0	0.01	0.2	0.2	0.22	4.1	23.0	0.01
grd-protocol	135.6	442.8	0.14	32.6	137.3	1.9	145.5	460.4	0.09
grd-psr	57.4	293.7	0.05	14.2	77.5	0.15	73.0	338.0	0.01
pipesworld	563.5	783.6	24.31	79.7	214.2	2.23	722.3	835.8	73.45
protocol	105.4	376.1	0.01	65.3	171.0	4.83	106.5	380.2	0.01
psr	9.1	69.5	0	12.9	44.1	3.3	9.0	70.1	0
satellite	321.2	646.2	1.64	35.4	120.7	0.38	527.8	779.4	16.02
IPC5	371.7	854.6	0.5	46.0	223.2	0.42	437.6	917.8	0.54
grd-pathways-prop	323.4	665.0	0.09	35.9	131.4	0.24	372.2	705.8	0.07
grd-openstacks	327.8	680.1	0.62	7.5	30.7	0.16	367.7	711.1	1.26
grd-pipesworld	329.0	666.2	1.36	224.6	645.2	3.05	335.4	667.1	1.28
grd-rover	253.0	595.4	0.22	13.2	35.2	0.54	309.3	648.8	0.21
grd-tpp-prop	246.5	585.9	0.31	16.1	57.0	0.1	314.3	650.6	0.44
grd-trucks	386.1	745.6	1.07	223.5	848.4	9.03	394.5	739.6	0.73
openstacks	191.5	518.7	0.05	28.8	97.1	0.2	231.5	570.1	0.01
pathways-prop	224.4	576.9	0.09	18.2	145.0	0.24	257.1	612.0	0.06
pipesworld	709.6	1414.9	21.38	97.6	234.9	4.74	827.4	1513.6	37.64
rover	372.4	696.3	0.89	27.1	128.8	0.51	549.7	795.6	9.55
storage-prop	403.2	714.7	0.58	31.0	137.4	0.05	546.8	791.8	13.08
tpp-prop	395.3	723.0	0.5	47.1	176.4	0.71	504.1	791.8	0.31
trucks	512.3	1422.2	0.05	56.2	180.7	0.18	546.1	1467.6	0.04
Hoffmann	54.7	297.9	0	4.6	53.6	0	110.3	422.1	0
assembly	1.1	5.8	0	0.0	0.0	0	1.2	6.0	0
bw-3ops	67.2	338.3	0	1.0	24.5	0	464.5	783.8	6.55
bw-4ops	34.4	226.7	0	4.0	48.8	0	190.5	523.5	0.13
briefcase	17.7	169.2	0	10.2	115.9	0	21.8	192.3	0
ferry	40.9	259.1	0	1.8	23.0	0	245.4	605.4	0.17
freecell	178.7	504.5	0.01	5.0	44.3	0	480.9	741.8	13.36
fridge	0.6	2.8	0	1.8	4.7	0	0.1	0.3	0
grid	87.8	379.1	0.01	3.6	36.5	0	486.4	791.1	13.42
gripper	67.0	334.4	0.01	1.8	10.5	0	279.7	650.9	0.93
hanoi	310.9	633.4	0.18	26.5	117.6	0.02	713.0	823.4	55.8
logistics	143.3	479.2	0	5.2	53.5	0	669.2	863.3	13.28
miconic-adl	1.8	46.6	0	2.9	6.2	0	1.5	51.4	0
miconic-simple	0.6	2.6	0	2.1	4.9	0	0.0	0.1	0
miconic	14.5	153.5	0	3.9	44.8	0	23.7	205.0	0
movie	42.8	234.6	0	11.1	91.0	0	100.9	368.3	0
mprime	131.1	459.5	0	11.1	82.5	0.01	213.6	578.9	0
mystery	76.2	353.8	0	9.1	87.5	0.01	155.5	502.6	0
schedule	14.4	148.0	0	7.5	60.9	0	16.7	167.1	0
tsp	1.9	10.8	0	1.8	11.4	0	2.1	9.1	0.05
tyreworld	200.0	564.0	0	0.0	0.0	0	373.3	737.7	0.12
Strict	74.3	338.1	0	3.4	37.9	0	112.8	414.0	0
art	1.1	3.9	0	0.7	2.9	0	2.2	5.8	0.02
art-1d	1.1	3.9	0	0.1	0.1	0	2.1	5.3	0
art-1d-rd	1.1	3.9	0	0.1	0.1	0	2.3	5.5	0
art-md	1.1	3.9	0	0.1	0.1	0	2.2	5.3	0
art-md-ns	1.1	3.9	0	0.1	0.1	0	2.3	5.5	0
art-md-ns-rd	7.8	107.5	0	0.4	3.5	0	16.8	159.5	0
art-md-rd	1.3	5.3	0	0.4	5.0	0	2.3	5.5	0
att-bw	0.8	3.4	0	2.0	5.4	0.03	0.0	0.1	0
att-bw2	0.6	2.5	0.03	0.2	0.4	0.04	1.4	4.2	0.01

Table 17 (continued)

Domain	All			Successes			Failures		
	mean	sd	med	mean	sd	med	mean	sd	med
att-logistics	198.9	556.9	0.01	8.8	40.5	0	458.6	786.9	0.02
bw	0.5	2.5	0	1.4	4.1	0	0.0	0.1	0
briefcase	0.5	2.4	0	0.7	2.9	0	0.1	0.2	0
briefcase-world	64.8	335.3	0	0.1	0.1	0.05	75.6	361.3	0
bulldozer	1.3	4.3	0	0.5	2.6	0	2.3	5.5	0
eight-puzzle	1.2	4.0	0	0.7	0.1	0.66	1.2	4.1	0
ferry	0.5	2.5	0	0.8	3.2	0	0.0	0.0	0
ferry-typed	0.5	2.5	0	0.8	3.1	0	0.0	0.0	0
flat-tire-adl	0.5	2.6	0	0	0.0	0	0.6	2.7	0
flat-tire	22.8	181.3	0	25.8	126.9	0.15	21.7	197.5	0
flat-tire-typing	22.0	195.7	0	1.0	3.6	0	26.0	213.5	0
fridge	32.7	240.4	0	0.1	0.2	0.12	33.9	244.8	0
fridge	36.2	240.2	0	1.5	4.7	0	60.4	312.6	0
homeowner	1.1	4.0	0	0.1	0.1	0.08	1.2	4.1	0
logistics	208.0	570.0	0.02	4.3	18.9	0.01	615.5	855.7	0.11
mcd-bw-axiom	64.6	336.0	0	–	–	–	64.6	336.0	0
meet-pass	129.4	464.9	0	0.6	2.6	0	198.7	565.6	0.02
molgen-adl	64.8	339.9	0	0.0		0.0	67.15	346.2	0
molgen	60.3	313.3	0	0.0	0.0	0	70.3	337.8	0
monkey	110.8	427.9	0	7.3	37.3	0	181.1	544.8	0
montlake	1.1	4.0	0	7.1	9.3	7.11	0.7	3.3	0
morris	33.2	240.3	0	0.0	0.1	0	43.3	274.2	0
movie	0.5	2.5	0	0.3	0.4	0.25	0.5	2.6	0
mystery	64.8	339.9	0	–	–	–	64.8	339.9	0
mystery	221.0	548.4	0.13	64.4	185.9	0.77	230.1	561.1	0.1
mystery-typed	218.4	549.1	0.07	58.5	148.7	1.62	227.2	561.6	0.05
people	0.5	2.5	0	0.1	0.2	0	0.6	2.7	0
prodigy-bw	48.9	291.7	0	0.2	0.7	0.01	91.0	395.2	0
road-operators	0.5	2.5	0	0.6	2.6	0	0.0	0	0
roads	33.2	240.3	0	0.9	3.3	0	46.2	284.4	0
safety-test1	0.5	2.5	0	3.4	6.5	0.13	0.0	0.1	0
safety-test2	0.5	2.5	0	3.4	6.5	0.14	0.0	0.1	0
simple-blocks	1.1	4.0	0	0.9	3.3	0	1.5	4.7	0
simple	46.5	284.3	0	1.8	1.8	1	47.9	288.7	0
snlp-bw1	16.6	170.0	0	0.9	3.2	0	40.0	268.2	0
snlp-bw2	48.7	291.7	0	0.1	0.4	0.01	106.9	427.3	0
trains	123.4	387.7	0.01	0.5	1.5	0.06	156.9	432.0	0
t-trains	64.9	335.8	0	0.3	0.7	0.01	79.0	369.5	0
uni-bw	0.5	2.5	0	0.1	0.1	0	0.6	2.6	0
woodshop	64.8	339.9	0	0.1	0.1	0.06	75.6	367.1	0
Sodor	243.9	601.5	0.01	35.1	121.0	0.15	287.5	650.9	0
three-way	243.9	601.5	0.01	35.1	121.0	0.15	287.5	650.9	0
STEK	248.6	604.2	0	8.5	17.2	1.44	259.1	615.1	0
stek	248.6	604.2	0	8.5	17.2	1.44	259.1	615.1	0

Appendix B. Planner descriptions

B.1. STRIPS and ADL planners

SystemR The planner R, written in Prolog, extends the STRIPS planner [2] in three key ways [35]. First, the planner stores multiple situations against which a potential action can be checked. Second, the planner handles subgoal cycles during search; for example, it will not add a subgoal such as "to achieve *g*, achieve *g* first". Third, it provides a way to achieve simple subgoals – goals that have no immediately applicable action – using either domain knowledge from the domain description or a straightforward strategy that selects an action that makes the subgoal true but may leave other goals open.

B.2. Partial order causal link planners

CPT-1.0 The CPT planner [36] implements powerful pruning techniques into an optimal temporal partial-order causal link paradigm. The key insight of this planner is to view search for an optimal plan as a constraint problem so as to harness a branch-and-bound style of search. The version we use is limited to canonical plans, which are plans in which every action is executed at most once.

PROD-4.0 The PRODIGY planning architecture [37,38] is a sound but incomplete bidirectional planner that integrates learning into the planning engine. Like UCPOP, this planner extends to some portions of the ADL; the planner handles conditional effects, negated preconditions, universal and existential preconditions. Search in PRODIGY proceeds by either regressing on (sub)goals or moving an action from the regressed "tail" to the "head" of the plan. At many of the decision points, PRODIGY uses learning to guide the search. We use a PRODIGY to PDDL language translator written by Eugene Fink.

SNLP-1.0 The **S**ystematic **N**on-**L**inear **P**lanner [39] performs IDA* searches of a lifted representation over partially ordered plans. But it searches by grounding a lifted operator in a non-deterministic choice. The planner also added the notion of causal links that allow one to state ordering constraints among actions in the plan that may threaten one another.

UCPOP-4.1 The **U**niversal quantification **C**onditional effects **P**artial **O**rder **P**lanner [40] combines the partial-order planning of SNLP with portions of Pednault's ADL. The planner handles conditional effects, quantified preconditions and effects, as well as universally quantified goals. The planning algorithm is both sound and complete. The version we use in this study was eventually superseded by the Sensory Graphplan (SGP) series of planners.

VHPOP-2.2 The **V**ersatile **H**euristic **P**artial **O**rder **P**lanner [41] provides a common implementation of the many strategies present in the literature while also adding strategies that incorporated the more recent advances of the distance-based heuristics frequently used in heuristic search planners. The search progresses using A* where the cost of the current plan is the number of actions and the heuristic cost is determined by the particular heuristic.

B.3. Graphplan planners

IPP-4.0 and IPP-4.1 The **I**nterference **P**rediction **P**lanner [42–44] extends the basic Graphplan algorithm to include ADL. IPP is sound and complete. Along with the description of the algorithm, the authors describe a method for a preprocessing step to convert ADL action schemas into STRIPS operators. The planner implements memoization of visited states in an efficient way and also includes a Goal Agenda Manager, which manages the order in which subgoals are achieved [45,46]. Version 4.1 reintroduced the RIFO engine – Reduction of Irrelevant Facts and Operators – which removes spurious details from the domain description and initial situation [47].

SGP-1.0b and SGP-1.0h The Sensory Graphplan planner [48] extends the basic Graphplan algorithm to include uncertainty. SGP extends Conformant Graphplan (CGP) by adding contingency in the final plan [49]. SGP implements a superset of UCPOP expressiveness but with a faster planner based upon Graphplan [48].

STAN-4 The **St**ate **An**alysis planner [50] increases the efficiency – in the form of bitvectors and a wave-front – of the Graphplan to speed up search. The bitvector representation, along with logical operators to apply actions and check mutexes, compresses the storage of the Graphplan while not changing search algorithm. The wave-front mechanism avoids explicitly constructing the graph beyond the point where no new facts are generated. STAN also includes the Type Interface Module (TIM) [27], for boosting performance in untyped domains.

B.4. Heuristic search planners

HSP-2.0, HSP-2.0r-h1plus, HSP-2.0r-h2max The **H**euristic **S**earch **P**lanner family [51] searches over the state-space of planning using heuristics that approximate the $h_{\max}$ heuristic by computing the costs of sets of atoms [52,53]. HSP2 performs best-first weighted-$A*$, where $W = 5$, search using an additive version of the relaxed Graphplan heuristic, h^1_{plus}, which is not admissible. We also included two variants of HSP2: HSP-2.0r-h1plus also uses the h^1_{plus} heuristic but searches backward. HSP-2.0r-h2max performs regressive search using the $h^2_{\max}$ heuristic that is equivalent to the Graphplan in parallel planning. A planner that is similar in many respects to the HSP family is TP4 [30].

FastDown The **F**ast **D**ownward family of planners [31,54] translates the planning problem into a multi-valued state representation, called SAS$^+$ [55], that is then mined for heuristic information. A key advantage of this system is that the translation to SAS$^+$ can identify structural dependencies in a Causal Graph – indeed, an early version of the planner was called the **C**ausal **G**raph planner. While the SAS formalism and heuristic search were both existing ideas, the contribution of this planner is to combine them in a planner that leverages the benefits of both.

FF-2.3 and Metric-FF The **F**ast **F**orward family of planners [56] uses enforced hill-climbing search over the planning state-space using an approximation of the h^+ heuristic. The planners handle ADL. Similar to the IPP family, the FF planners assume potential independence of subgoals and operate under a goal agenda determined by a preprocessing step. The enforced hill-climbing algorithm is guaranteed to find a solution if the planning problem is dead-end free; if enforced hill-climbing fails, then the planner switches to greedy best-first search. Two other important features are its emphasis on helpful actions, which add at least one goal for the next step, and added goal deletion, which identifies goals that are achieved too early. A metric version of the planner [57] extends the key ideas of its predecessor to handle temporal domains *during* search.

LPG-1.1, LPG-1.2, LPG-TD The **L**ocal Search on **P**lanning **G**raph family of planners [29,58–60] are among the few planners that incorporate alternative metrics *during* search. The planner works by performing stochastic local search on planning graph subsets, which are partial plans that the authors call action graphs. At each branch in the search, the choice to add or remove an action is guided by a heuristic that rates each potential choice with respect to the

current partial solution; the heuristic function for each potential action is the count of unsupported facts plus the count of actions which are mutex. The authors add a number of guidance mechanisms including weighting the heuristic, weighting actions with Lagrange multipliers, and performing promotion and demotion similar to that found in POCL planners. The planner was extended to handle PDDL 2.1 [61] and PDDL 2.2 [62]. The latest version of the planner uses many additional techniques beyond those of the original techniques for planning graphs.[11]

OPTOP The Optop planner [63,64] is a regression planner that searches over state spaces. It uses regression match graphs [65,66] to select among its choices which action to apply to the current state. It is one of the few planners that incorporates estimates of quality *during* search. Notably, Optop handles the full ADL.

SimPlan-2.0 The SimPlan planner [67] was designed to work in a interleaved (simulated) planning and execution framework; thus, many of the design decisions focused on creating a fast replanner that could return a valid, but sub-optimal, plan. The planner constructs a relaxed Graphplan in a forward phase then uses a backtrack-free "BackwardGraph" phase that constructs partial solutions called Approximate Plans aggregated into a tree, from which it then searches by examining potential goal interactions within the Approximate Plans.

SAPA-2 The SAPA planner [68] is another planner that uses metric information *during* search. The planner is a forward chaining heuristic search planner that uses A* search over time stamped states, which include the situation plus potential future events that have not yet occurred. The planner uses a complex notion of temporal considerations during planning. It constructs its heuristics from the relaxed temporal Graphplan.

B.5. Hybrid planners

AltAlt-1.0 **A L**ittle of **T**his, **A L**ittle of **T**hat [69,70] combines STAN 3.0 and HSP-r to gain the advantages of the strong heuristics and CSP-style search that are provided with Graphplan with the strength of regression search over the planning state space. A key contribution of the planner is in examining a number of principled extensions to the heuristics that can be used. The more recent version added partial construction of the planning graph and limits the branching factor of regression search by using subsets of actions.

SGPlan-06 SGPlan [71,72] uses Lagrangian multipliers between subplans constructed by partitioning on the subgoals. The planner uses a number of techniques to improve its search, including an enhanced path-finding algorithm for identifying landmarks that aid in plan decomposition.

B.6. SAT/BDD encoding planners

BlkBox-4.2, Satplan04, Satplan06 The SATPlan/BlackBox family of planners transform the original PDDL representation into a SAT representation. The original planner [73] used a direct translation, while subsequent models [74] also incorporated an intermediate transformation using the Graphplan. The latest version [75] allows both action-based encoding and fluent encoding into propositional formulae and further extended the pruning techniques.

MIPS-3 The **I**ntelligent **M**odel-checking and **P**lanning **S**ystem [76] searches over the space of Boolean formulas; however, MIPS searches bidirectionally using Binary Decision Diagrams for its representation. MIPS-3 uses a closed list to prevent cycling, simplifies successor sets, and employs the symbolic version of A*, called BDDA*.

References

[1] A. Blum, M. Furst, Fast planning through planning graph analysis, in: Proc. of the 14th Int'l Joint Conf. on Artificial Intelligence (IJCAI-95), Montreal, Quebec, Canada. August 20–25, 1995, pp. 1636–1642.
[2] R.E. Fikes, N.J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, Artificial Intelligence 2 (3–4) (1971) 189–208.
[3] E. Pednault, ADL: Exploring the middle ground between strips and the situation calculus, in: R.J. Brachman, H.J. Levesque, R. Reiter (Eds.), Proc. of the 1st Int'l Conf. on Principles of Knowledge Representation and Reasoning (KR-89), Toronto, Canada, May 15–18, 1989, Morgan Kaufmann, 1989.
[4] AIPS-98 International Planning Competition Committee, PDDL: The planning domain definition language, Tech. rep., Yale Center for Computational Vision and Control, the AIPS-98 committee consisted of: M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, D. Wilkins, October 1998.
[5] J. Hoffmann, S. Edelkamp, The deterministic part of IPC-4: An overview, Journal of Artificial Intelligence Research 24 (2005) 519–579.
[6] D. Long, M. Fox, The 3rd international planning competition: Results and analysis, Journal of Artificial Intelligence Research 20 (2003) 1–59.
[7] A. Howe, E. Dahlman, A critical assessment of benchmark comparison in planning, Journal of Artificial Intelligence Research 17 (2002) 1–33.
[8] J. Hoffmann, Utilizing Problem Structure in Planning: A Local Search Approach, Springer-Verlag, Berlin, New York, 2004.
[9] D. McDermott, The 1998 AI planning systems competition, AI Magazine 2 (2) (2000) 35–55.
[10] F. Bacchus, AIPS'00 planning competition, AI Magazine 22 (3) (2001) 47–56.
[11] Y. Dimopoulos, A. Gerevini, P. Haslum, A. Saetti, The benchmark domains of the deterministic part of IPC-5, in: Special Booklet for the Planning Competition in the Working Notes of the 16th Int'l Conf. on Automated Planning and Scheduling (ICAPS-06), Monterey, CA, USA, June 5–10, 2006.
[12] A. Howe, E. Dahlman, C. Hansen, M. Scheetz, A. vonMayrhauser, Exploiting competitive planner performance, in: S. Biundo, M. Fox (Eds.), Proc. 5th European Conf. on Planning (ECP-99), Durham UK September 8–10, in: Lecture Notes on Computer Science, vol. 1809, Springer, 1999.
[13] R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0, 2006.

[11] Personal communication.

[14] J. Hoffmann, Local search topology in planning benchmarks: An empirical analysis, in: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01), Kaufmann, Seattle WA, 2001, pp. 453–458.
[15] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed., Morgan Kaufmann, San Francisco, 2005.
[16] J. Hoffmann, Local search topology in planning benchmarks: A theoretical analysis, Tech. Rep. 165, Institute for Computer Science, Albert Ludwigs University, Georges-Koehler-Allee, Geb. 52, 79110 Freiburg, Germany, November 16 2001.
[17] J. Hoffmann, Where ignoring delete lists works: Local search topology in planning benchmarks, Journal of Artificial Intelligence Research 24 (2005) 685–758.
[18] E. Taillard, Benchmarks for basic scheduling problems, European Journal of Operational Research 64 (1993) 278–285.
[19] D.E. Wilkins, Using the SIPE-2 Planning System: A Manual for Version 6.1, SRI International, Artificial Intelligence Center, Menlo Park, CA, 1999.
[20] J. Watson, L. Barbulescu, A. Howe, L. Whitley, Algorithm performance and problem structure for flow-shop scheduling, in: Proc. of the 16th National Conference on Artificial Intelligence (AAAI-99), Orlando, FL, AAAI/MIT Press, Anaheim, CA, 1999.
[21] W. Ruml, M.B. Do, M.P. Fromherz, On-line planning and scheduling for high-speed manufacturing, in: Proc. of the 15th Int'l Conf. on Automated Planning and Scheduling (ICAPS-05), Monterey, CA, USA, June 5–10, 2005, AAAI Press, Menlo Park, CA, 2005, pp. 30–39.
[22] M. Haitjema, Y. Chen, C. Roman, C. Gill, A PDDL domain generator for workflow allocation, available at http://www.cse.wustl.edu/~chen/workflow/, 2007.
[23] M. Boddy, J. Gohde, J.T. Haigh, S. Harp, Course of action generation for cyber security using classical planning, in: Proc. of the 15th Int'l Conf. on Automated Planning and Scheduling (ICAPS-05), Monterey, CA, USA, June 5–10, 2005, AAAI Press, Menlo Park, CA, 2005.
[24] M. Veloso, J. Blythe, Linkability: Examining causal link commitments in partial-order planning, in: K.J. Hammond (Ed.), Proc. of the 2nd Int'l Conf. on Artificial Intelligence Planning Systems (AIPS-94), Chicago, Illinois, June 13–15, 1994, AAAI Press, 1994, pp. 170–175.
[25] M. Roberts, A. Howe, B. Wilson, M. desJardins, What makes planners predictable? in: Proc. of the 18th Int'l Conf. on Automated Planning and Scheduling (ICAPS-08), Sydney, Australia, September 14–18, 2008, AAAI Press, Menlo Park, CA, 2008.
[26] D. Vrakas, G. Tsoumakas, N. Bassiliades, I. Vlahavas, Machine learning for adaptive planning, in: Intelligent Techniques for Planning, Idea Group, 2005, pp. 90–120.
[27] M. Fox, D. Long, The automatic inference of state invariants in TIM, Journal of Artificial Intelligence Research 9 (1998) 367–421.
[28] Y. Chen, X. Zhao, W. Zhang, Long distance mutual exclusion for propositional planning, in: Proc. of the 20th Int'l Joint Conf. on Artificial Intelligence (IJCAI-07). Hyderabad, India, January 6–12, 2007.
[29] A. Gerevini, I. Serina, Planning as propositional CSP: from Walksat to local search for action graphs, CONSTRAINTS 8 (2003) 389–413.
[30] P. Haslum, Improving heuristics through relaxed search – an analysis of TP4 and HSP*a in the 2004 planning competition, Journal of Artificial Intelligence Research 25 (2006) 233–267.
[31] M. Helmert, The fast downward planning system, Journal of Artificial Intelligence Research 26 (2006) 191–246.
[32] A.I. Coles, A.J. Smith, Marvin: A heuristic search planner with online macro-action learning, Journal of Artificial Intelligence Research 28 (2007) 119–156.
[33] M. Helmert, Complexity results for standard benchmark domains in planning, Artificial Intelligence 143 (2) (2003) 219–262.
[34] M. Helmert, New complexity results for classical planning benchmarks, in: Proc. of the 16th Int'l Conf. on Automated Planning and Scheduling (ICAPS-06), Cumbria, UK, June 6–10, 2006, AAAI Press, Menlo Park, CA, 2006, pp. 52–61.
[35] F. Lin, A planner called R, Artificial Intelligence 22 (3) (2001) 73–76.
[36] V. Vidal, H. Geffner, Branching and pruning: An optimal temporal POCL planner based on constraint programming, in: Proc. of the 19th National Conference on Artificial Intelligence (AAAI-04), San Jose, CA, 2004.
[37] E. Fink, J. Blythe, Prodigy bidirectional planning, Journal of Experimental and Theoretical Artificial Intelligence 17 (3) (2005) 161–200.
[38] M. Veloso, J. Carbonell, A. Perez, D. Borrajo, E. Fink, J. Blythe, Integrating planning and learning: The PRODIGY architecture, Journal of Experimental and Theoretical Artificial Intelligence 7 (1) (1995) 81–120.
[39] D. McAllester, D. Rosenblitt, Systematic nonlinear planning, in: Proc. of the 9th National Conference on Artificial Intelligence (AAAI-91), vol. 2, AAAI Press/MIT Press, Anaheim, CA, 1991, pp. 634–639.
[40] J.S. Penberthy, D. Weld, UCPOP: A sound, complete, partial-order planner for ADL, in: B. Nebel, C. Rich, W.R. Swartout (Eds.), Proc. of the 3rd Int'l Conf. on Principles of Knowledge Representation and Reasoning (KR-92), Cambridge, MA, October 25–29, 1992, Morgan Kaufmann, 1992.
[41] H. Younes, R. Simmons, VHPOP: Versatile heuristic partial order planner, Journal of Artificial Intelligence Research 20 (2003) 405–430.
[42] J. Koehler, B. Nebel, J. Hoffmann, Y. Dimopoulos, Extending planning graphs to an ADL subset, Tech. Rep. 88/97, Institute for Computer Science, Albert Ludwigs University, 1997.
[43] J. Koehler, B. Nebel, J. Hoffmann, Y. Dimopoulos, Extending planning graphs to an ADL subset, in: S. Steel, R. Alami (Eds.), Proc. of the 4th European Conference on Planning (ECP-97), Toulouse, France, September 24–26, 1997, in: Lecture Notes in Computer Science, vol. 1348, Springer, 1997, pp. 273–285.
[44] J. Koehler, IPP homepage, Web Page at http://www.informatik.uni-freiburg.de/~koehler/ipp.html, last accessed September 11, 2007 (October 15 2006).
[45] J. Koehler, J. Hoffmann, Planning with goal agendas, Tech. Rep. 110, Institute for Computer Science, Albert Ludwigs University, 1998.
[46] J. Koehler, J. Hoffmann, On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm, Journal of Artificial Intelligence Research 12 (2000) 338–386.
[47] B. Nebel, Y. Dimopoulos, J. Koehler, Ignoring irrelevant facts and operators in plan generation, in: Proceedings of the 4th European Conference on Planning, in: LNAI, vol. 1348, Springer, 1997, pp. 338–350.
[48] D.S. Weld, C.R. Anderson, D.E. Smith, Extending Graphplan to handle uncertainty and sensing actions, in: Proc. of the 15th National Conference on Artificial Intelligence (AAAI-98), Madison, WI, AAAI/MIT Press, Anaheim, CA, 1998, pp. 897–904.
[49] D.S. Weld, Recent advances in AI planning, AI Magazine 20 (2) (1999) 93–123.
[50] D. Long, M. Fox, Efficient implementation of the plan graph in STAN, Journal of Artificial Intelligence Research 10 (1999) 87–115.
[51] B. Bonet, H. Geffner, Planning as heuristic search, Artificial Intelligence 129 (1–2) (2001) 5–33.
[52] B. Bonet, H. Geffner, Planning as heuristic search: New results, in: S. Biundo, M. Fox (Eds.), Proc. 5th European Conf. on Planning (ECP-99), Durham, UK, September 8–10, 1999, in: Lecture Notes on Computer Science, vol. 1809, Springer, 1999, pp. 359–371.
[53] P. Haslum, H. Geffner, Admissible heuristics for optimal planning, in: Proc. of the 5th Int'l Conf. on Artificial Intelligence Planning Systems (AIPS-00), Breckenridge, CO, USA, April 14–17, 2000, AAAI Press, Menlo Park, CA, 2000, pp. 140–149.
[54] M. Helmert, A planning heuristic based on causal graph analysis, in: Proc. of the 14th Int'l Conf. on Automated Planning and Scheduling (ICAPS-04), Whistler, BC, Canada, AAAI Press, Menlo Park, CA, 2004, pp. 161–170.
[55] C. Bäckström, B. Nebel, Complexity results for SAS^+ planning, Computational Intelligence 11 (4) (1995) 625–655.
[56] J. Hoffmann, B. Nebel, The FF planning system: Fast plan generation through heuristic search, Journal of Artificial Intelligence Research 14 (2001) 253–302.
[57] J. Hoffmann, The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables, Journal of Artificial Intelligence Research 20 (2003) 291–341.
[58] A. Gerevini, I. Serina, LPG: A planner based on local search for planning graphs, in: M. Ghallab, J. Hertzberg, P. Traverso (Eds.), Proc. of the 6th Int'l Conf. on Artificial Intelligence Planning Systems (AIPS-02), Toulouse, France, April 23–27, 2002, AAAI Press, Menlo Park, CA, 2002, pp. 13–22.

[59] A. Gerevini, I. Serina, A. Saetti, S. Spinoni, Local search techniques for temporal planning in LPG, in: Proc. of the 13th Int'l Conf. on Automated Planning and Scheduling (ICAPS-03), Trento, Italy, June 9–13, 2003, AAAI Press, 2003.
[60] A. Gerevini, A. Saetti, I. Serina, An approach to temporal planning and scheduling in domains with predicatable exogenous events, Journal of Artificial Intelligence Research 25 (2006) 187–231.
[61] A. Gerevini, A. Saetti, I. Serina, Planning through stochastic local search and temporal action graphs in LPG, Journal of Artificial Intelligence Research 20 (2003) 239–290.
[62] A. Gerevini, A. Saetti, I. Serina, Planning in PDDL2.2 domains with LPG-TD, in: Special Booklet for the Planning Competition in the Working Notes of the 14th Int'l Conf. on Automated Planning and Scheduling (ICAPS-04), Whistler, BC, Canada, 2004.
[63] D. McDermott, The Optop planner, in: Special Booklet for the Planning Competition in the Working Notes of the 16th Int'l Conf. on Automated Planning and Scheduling (ICAPS-06), Monterey, CA, USA, June 5–10, July 2004.
[64] D. McDermott, The Opt and Optop API, 0.9 edition, Yale Computer Science Department, 2005.
[65] D. McDermott, A heuristic estimator for means ends analysis in planning, in: B. Drabble (Ed.), Proc. of the 3rd Int'l Conf. on Artificial Intelligence Planning Systems (AIPS-98), Edinburgh, Scotland, May 29–31, 1996, AAAI Press, Menlo Park, CA, 1996, pp. 142–149.
[66] D. McDermott, Using regression-match graphs to control search in planning, Artificial Intelligence 109 (1-2) (1999) 111–159.
[67] O. Sapena, E. Onainda, Domain independent on-line planning for STRIPS domains, in: Proc. IBERAMIA-02, in: LNAI, vol. 2527, Springer-Verlag, 2002.
[68] M.B. Do, S. Kambhampati, Sapa: A multi-objective metric temporal planner, Journal of Artificial Intelligence Research 20 (2003) 155–194.
[69] R. Sanchez-Nigenda, X. Nguyen, S. Kambhampati, AltAlt: Combining the advantages of graphplan and heuristic state search, in: Proceedings of the 3rd International Conference on Knowledge-based Systems (KBCS), Bombay, 2000.
[70] X. Nguyen, S. Kambhampati, R.S. Nigenda, Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search, Artificial Intelligence 135 (2002) 73–124.
[71] Y. Chen, B.W. Wah, Automated planning and scheduling using calculus of variations in discrete space, in: Proc. of the 13th Int'l Conf. on Automated Planning and Scheduling (ICAPS-03), Trento, Italy, June 9–13, 2003, AAAI Press, 2003, pp. 2–11.
[72] Y. Chen, B.W. Wah, C.-W. Hsu, Temporal planning using subgoal partitioning and resolution in SGPlan, Journal of Artificial Intelligence Research 26 (2006) 323–369.
[73] H. Kautz, B. Selman, BLACKBOX: A new approach to the application of theorem proving to problem solving, in: Workshop on Planning as Combinatorial Search, Working Notes of the 4th Int'l Conf. on Artificial Intelligence Planning Systems (AIPS-98), Pittsburgh, PA, USA, June 7–10, 1998, AAAI Press, Menlo Park, CA, 1998.
[74] H. Kautz, B. Selman, Unifying SAT-based and graph-based planning, in: Proc. of the 16th Int'l Joint Conf. on Artificial Intelligence (IJCAI-99), Stockholm, Sweden, July 31–August 6, 1999.
[75] H. Kautz, B. Selman, J. Hoffmann, Satplan: Planning as satisfiability, in: Special Booklet for the International Planning Competition in the Working Notes of the 16th Int'l Conf. on Automated Planning and Scheduling (ICAPS-06), Monterey, CA, USA, June 5–10, 2006.
[76] S. Edelkamp, M. Helmert, On the implementation of MIPS, in: Proc. of the 5th Int'l Conf. on Artificial Intelligence Planning Systems (AIPS-00), Breckenridge, CO, USA, April 14–17, 2000, AAAI Press, Menlo Park, CA, 2000.