Performance Modelling of Planners from Homogeneous Problem Sets

Tomás de la Rosa, Isabel Cenamor, and **Fernando Fernández**
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. Leganés (Madrid). Spain
icenamor@inf.uc3m.es,trosa@inf.uc3m.es, ffernand@inf.uc3m.es

Abstract

Empirical performance models play an important role in the development of planning portfolios that make a per-domain or per-problem configuration of its search components. Even though such portfolios have shown their power when compared to other systems in current benchmarks, there is no clear evidence that they are capable to differentiate problems (instances) having similar input properties (in terms of objects, goals, etc.) but fairly different runtime for a given planner. In this paper we present a study of empirical performance models that are trained using problems having the same configuration, with the objective of guiding the models to recognize the underlying differences existing among homogeneous problems. In addition we propose a set of new features that boost the prediction capabilities under such scenarios. The results show that the learned models clearly performed over random classifiers, which reinforces the hypothesis that the selection of planners can be done on a per-instance basis when configuring a portfolio.

Introduction

Empirical Performance Models (EPMs) for automated planners are built to predict the behavior of planners when they solve a particular planning task. The straightforward application of these EPMs is the per-instance selection of a good planner or set of candidate planners for solving the task. Other interesting application is the generation of hard planning tasks without actually running any planner. EPMs should be trained extracting from the learning instances a set of relevant features able to discriminate, given a specific planner, between its different performance outcomes. However, there is no clear evidence that EPMs are able to classify planning tasks by their intrinsic difficulty and not by other properties that make them different from other examples of the training set: on the one hand, many features used to characterize planning tasks are considered weak given that they might not have a direct correlation with the task difficulty; on the other hand, when input files are similar, the vast majority of the state of the art features extracted from the instantiation of the tasks produce the same or fairly similar values, and therefore they do not add discriminant information to recognize which tasks are potentially difficult for a given planner. From our point of view, given a particular planner, an EPM may encode knowledge as a combination of the following capabilities:

- *Domain discrimination*: the models learn how to recognize planning tasks from a certain domain, usually using features at domain/problem level (i.e., features generated from the PDDL (*Planning Domain Definition Language*) input files. Then, the models tend to predict the average performance seen on that domain. In this case, we can consider that these models are over-fitted, given that there is no justified correlation between shallow features such as the number of PDDL actions and the difficulty of a planning task. Moreover, consider that most of these features could be artificially modified in the PDDL description without altering the performance of a planner at all (e.g., adding a number of actions that never become applicable, but that alter the feature values).
- *Size discrimination*: the models give relevance to features from the instantiation, so they can recognize the magnitude or the size of the problem. This makes sense because, for a given domain, larger problems would need more time to run on average. This effect is particularly dominant when models are trained with problems of a single domain.
- *Search space discrimination*: The models predict the performance based on features from some characteristics that are intrinsic to the search tree or to the process of exploring it. This kind of discrimination could be the only one useful when trying to classify problems of the same size and configuration.

Planning EPMs have been usually trained using a set of available benchmarks. Most of these benchmarks have been created for the evaluation in *International Planning Competitions* (IPCs). Thus, each problem set has planning tasks of incremental size to show how state-of-the-art planners scale in their performance until hopefully fail in the largest instances. Under these circumstances is very hard to isolate the effect of different discrimination types. For the case of the learning tracks of the IPC, the models are trained with problem sets of the same domain, and therefore only the size and and the search discrimination could be combined into the model. In other tracks, like deterministic, domain and size discrimination would be enough to obtain very high prediction performance.

The contributions of this work are: (1) The empirical study that confirms that EPMs for planning are able to show the search space discrimination. We achieve this target by training EPMs with homogeneous problem sets (i.e., preventing the models from using other discrimination capabilities) and validating that these models perform better than random classifiers. (2) A new set of easily computable features that improve prediction capabilities of state-of-the-art EPMs for planning. (3) An analysis of the feature relevance to recognize which features are important to predict the empirical hardness of planning task when they belong to homogeneous sets.

Related Work

EPMs of automated planners have evolved during the last years. Roberts et al. 2008; 2009 trained EPMs on known benchmarks up to 2008, showing good accuracy on predicting whether a planner will succeed or not. The features used to generate their models were from the domain and problem definition. Therefore, this set of features is not useful for models trained with homogenous problem sets given that they will produce the same values. Cenamor et al. 2012; 2013 trained EPMs including features from the causal graph and domain transition graphs, as well as some instantiation level features extracted from the Fast-Downward translator (Helmert 2006). These features contributed to the learning of both classification and regression models: a classification model was able to predict whether a given planner will solve an input problem in the given time (1800 seconds) and, if so, the regression model was able to estimate how long it would take to achieve the best solution. Nevertheless, these models were trained using existing benchmarks, which contains problems of incremental size, and therefore can not give a clear evidence on the effect of the search space discrimination. Fawcett et al. 2014 extended the set of Cenamor's features including features extracted by probing the search with short runs of Fast Downward, and features extracted from the SAT encoding of the planning task. Results showed they improved existing regression models. However, the SAT-encoding features were reported as computationally expensive, and therefore less attractive to include in EPMs generated for a per-instance selection of planners in a portfolio. Moreover, as these features are not expressed in terms traditional planning task elements, it seems difficult to analyze results if one want to recognize what makes a planning task easy or hard. As other work, these EPMs were trained with existing benchmarks. Fawcett et al. also included features from Torchlight (Hoffmann 2011), which computes features under the topology of delete relaxation heuristics and its relation to causal graphs. However they reported that Torchlight features were in general less relevant for such predictions. The IBACOP2 planner (Cenamor, de la Rosa, and Fernández 2016), the winner of the sequential satisficing track of IPC-2014, is a planning portfolio that uses EPMs to make a per-instance selection of planners to run. It extends the set of features of a previous work (Cenamor, de la Rosa, and Fernández 2013) but rather than probing the search, it includes heuristic values of the initial state using different heuristics in the planning literature. In addition, IBACOP2 includes fact balance features, a set features extracted from the relaxed plan of the initial state that enrich its characterization in addition to the single computation of the FF heuristic (Hoffmann and Nebel 2001). As the rest of research work, IBACOP2 models were trained using existing benchmarks, therefore even that they claim to make a per-instance configuration of the portfolio, from their result we can not conclude that these results are supported by the search space discrimination. In this work we have used the set of features included in IBACOP2 as our starting point to evaluate whether the search space discrimination exists and to propose additional features.

Planning Task Features

We have augmented the set of features proposed in IBACOP2 by including (1) the Red-Black heuristic (Domshlak, Hoffmann, and Katz 2015) to the list of heuristic values computed for the initial state; (2) new fact balance features that give an overall measure to characterize the relaxed plan rather than giving statistics of its elements and (3) features extracted from the landmark graph computed also in the initial state. The last two groups are explained in detail in this section.

Fact Balance Features

As background knowledge for describing fact balance features we state that a planning task is defined as the tuple $\Pi = \langle \mathcal{P}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{P}$ is a set of propositions, $\mathcal{A}$ is a set of actions, $\mathcal{I}$ is the initial state and $\mathcal{G}$ is the set of goals. Each $a \in \mathcal{A}$ is defined as the tuple $\langle pre(a), add(a), del(a) \rangle$ representing the action preconditions, added effects and deleted effects respectively. A plan $\pi = \{a_1, \dots, a_n\}$ is a sequence of actions that transforms $\mathcal{I}$ into a new state where goals in G have been achieved.

A relaxed plan π^+ is the sequence of actions that solves a relaxed planning task, i.e. a task in which deleted effects of actions have been ignored (Hoffmann and Nebel 2001). The plan $\pi^\pm$ is defined as the sequence of actions corresponding to π^+, but with actions taken from the original task. This plan is obviously not applicable in most of the cases, but it provides useful information for a variety of search control techniques, such as lookahead states (Vidal 2004). For the fact balance, $\pi^\pm$ is the base for encoding additional information that is not incorporated in the heuristic values of delete relaxation heuristics. Particularly, the information is computed from the relaxed plan of the initial state, which is denoted by $\pi^\pm_\mathcal{I}$. We will still call $\pi^\pm$ a relaxed plan to remark it is basically obtained by the original procedure of π^+.

Definition 1 *Given a relaxed plan* $\pi^\pm$, **the balance of a fact** *p is a function $\mathcal{B}$ that computes the difference of the number of times p is added and the number of times p is deleted in* $\pi^\pm$.

$$\mathcal{B}(p, \pi^\pm) = |\{a | a \in \pi^\pm \wedge\ p \in add(a)\}| - |\{a | a \in \pi^\pm \wedge\ p \in del(a)\}| \quad (1)$$

The idea behind computing the balance of facts is that we can reflect with a number whether the relaxation is altering more or less the possible application of a sequence of actions from $\pi^{\pm}$. A negative balance for a fact p indicates that $\pi^{\pm}$ probably will not be applicable, and p will have to be recovered by other actions in the real plan.

From a propositional machine learning perspective, one would like to generate a fixed set of features that characterize a relaxed plan. However, the number of propositions is dependent of the planning task, therefore the balance of facts can not be encoded as individual features. Instead, the information is aggregated computing statistics over set of propositions. The set of fact balance features considered in IBACOP2 are:

- The min., average, and standard deviation of $\mathcal{B}(p, \pi^{\pm}_{\mathcal{I}}), \forall p \in \mathcal{P}$.
- The min., average, and standard deviation of $\mathcal{B}(g, \pi^{\pm}_{\mathcal{I}}), \forall g \in \mathcal{G}$.

For this work we have generated three new features that also provide additional information regarding the *Relaxed Planning Graph* (RPG) built by the FF planner (Hoffmann and Nebel 2001). An RPG is a sequence of proposition and action layers $P_0, A_0, P_1, A_1, \ldots, A_{t-1}, P_t$, representing the reachability analysis of the relaxed planning task. Each action in $\pi^{\pm}$ has an associated layer that we denote with $level(a)$. The FF heuristic is the number of actions in the relaxed plan. However, this number does not reflect many properties of the structure of $\pi^{\pm}$ and its corresponding RPG. The idea behind these new features is to partially encode this underlying structure, and therefore find the difference between tasks that may have the same heuristic value in the initial state.

Definition 2 **the level balance of a fact** p **at layer** l *is a function* $\mathcal{B}_{[l]}$ *that computes the difference of the number of times* p *is added and the number of times* p *is deleted by actions of* $\pi^{\pm}$ *that belongs to layer* $l-1$.

$$\mathcal{B}_{[l]}(p, \pi^{\pm}) = \mathcal{B}(p, \{a | a \in \pi^{\pm} \wedge \ level(a) = l-1\})$$

In the same way, we define the function $\mathcal{B}_{[*l]}(p, \pi^{\pm})$ to compute the **sum of level balance for a fact up to layer** l, which considers the prefix of $\pi^{\pm}$ that includes all actions from layer 0 to $l-1$.

We compute the balance in each RPG layer because we can use it to realize part of the structure of $\pi^{\pm}$. If we imagine the application of $\pi^{\pm}$ by a group of actions (i.e, divided by each layer), each layer would have a mark or a "*balance footprint*" of how far is $\pi^{\pm}$ from being applicable. Figure 1 presents the *BalanceFootprint* function, an algorithm that computes the following values:

1. fp^{+}_{l}, the balanced footprint in layer l, as a measure that aggregates the occurrences in which a fact has a positive balance.
2. fp^{-}_{l}, the unbalanced footprint in a layer l, as a measure that aggregates the occurrences in which a fact has a negative balance.
3. $dist\_fp_l$, the distortion of the unbalanced footprint to record whether the facts remain unbalanced for many layers. This measure is computed as an exponential penalty of the number of layers, to represent that unbalanced facts tends to produce uninformed heuristic values specially if this situation holds for many layers or until RPG fix-point.

```
BalanceFootprint ($RPG, \pi^{\pm}, l$): ($fp^{+}, fp^{-}, dist\_fp$)
----------------------------------------------------------------
$fp^{+}$: (positive) balanced footprint of a layer
$fp^{-}$: (negative) unbalanced footprint of a layer
$dist\_fp$ : distortion of the unbalanced footprint
----------------------------------------------------------------
$fp^{+} = 0; fp^{-} = 0; dist\_fp = 0$
for each $p$ in $\mathcal{P}$ do
    if $is\_goal(p)$ then
        target = 1
    else
        target = 0
    if $\mathcal{B}_{[l]}(p, \pi^{\pm})$ =! 0 then
        if $\mathcal{B}_{[*l]}(p, \pi^{\pm}) \geq$ target then
            $fp^{+}$ += $\mathcal{B}_{[*l]}(p, \pi^{\pm})$ - target
        else
            $fp^{-}$ += target - $\mathcal{B}_{[*l]}(p, \pi^{\pm})$
            lgap = 1
            diff = $\mathcal{B}_{[*l]}(p, \pi^{\pm})$
            while ((lgap+$l$) < $layers(RPG)$ and diff $\geq$ target)
                diff += $\mathcal{B}_{[l]}(p, \pi^{\pm})$
                lgap += 1
            $dist\_fp$ += (target - $\mathcal{B}_{[*l]}(p, \pi^{\pm})$) * $2^{\mathrm{lgap}}$
```

Figure 1: Algorithm for computing the positive and negative balance footprints for a layer of the RPG.

The balance footprints computed by the algorithm in Figure 1 are associated to particular layers of RPG. Also, their magnitude is affected by the number of actions in the previous layer. Therefore, to compute the balance and unbalance features in a normalized way, we aggregate the value of each layer multiplying it by a weight that represent the proportion of actions that appear in each particular layer of RPG. The distorsion measure involve information of several layers, therefore it will not be weighted. Thus, after computing the balance footprints, the three new features are computed as follows:

$$BalancedRP = \sum_{i=1}^{layers(RPG)} \frac{|A_{i-1}|}{|\mathcal{A}|} \times fp^{+}_{i}$$

$$UnbalancedRP = \sum_{i=1}^{layers(RPG)} \frac{|A_{i-1}|}{|\mathcal{A}|} \times fp^{-}_{i}$$

$$DistortedRP = \sum_{i=1}^{layers(RPG)} dist\_fp_i$$

Landmark Graph Features

The landmarks of a planning task are a set of propositions that have to become true at some point in every solution plan. They give information about specific conditions that must hold in intermediate steps of the search process. Nevertheless, as many of them can be computed beforehand in a pre-processing step, this knowledge is suitable for characterizing planning tasks from a non-syntactical perspective. The set of landmarks and their orderings induce a landmark graph. In this work, we have used the procedure provided by Fast-Downward to extract landmarks, which creates a single graph merging two different techniques for extracting landmarks (Richter, Helmert, and Westphal 2008; Zhu and Givan 2003). We use the landmark graph to compute the new set of features proposed next:

- Number of landmarks (nodes in the landmark graph)
- Number of edges in the landmark graph
- Number of parent nodes (i.e., landmarks with no input edges)
- Number of children nodes (i.e., landmarks with no output edges)
- Number of inner nodes
- The ratio between landmarks and edges
- Maximum, average and standard deviation of the number of input edges for a landmark
- Maximum, average and standard deviation of the number of output edges for a landmark

The first four features are new to the set of IBACOP2 features, but should not be considered as new features in the literature given that they were included as part of Fast Downward search probe features proposed by Fawcett et al. 2014.

Experimental Setup

In this section we describe the methodology we have followed to prepare the experimental evaluation for learning EPMs from homogeneous problem sets. We say that two problems of a given domain are homogeneous if:

1. they have the same distribution of objects.
2. their initial state and goals have been sampled from the same state/goals distributions

Even though the second condition might seem open to cover fairly different problems, what we want to state is that we are interested in problems that produce a zero or near zero variance for features generated from a syntactical pre-process of the input files. From a practical point of view, we consider that a problem set is homogeneous if their problems have been generated with the same input parameters of a random problem generator.

Next, we enumerate the planners selected and describe how we have selected the planning domains and problems; then, we motivate the performance metric used in the learning evaluation and describe how we have created the training sets.

Planners, Domains and Problems

For the evaluation we have selected three stand alone planners that have shown great performance on the sequential satisficing tracks of recent IPCs and that are reasonably different in their search techniques. The selected planners are: LAMA (Richter, Helmert, and Westphal 2008; Richter, Westphal, and Helmert 2011), the winner of IPC-2011; MERCURY (Domshlak, Hoffmann, and Katz 2015) the third best planner in terms of quality score and the best stand-alone planner in IPC-2014; and PROBE (Lipovetzky and Geffner 2011), the second best planner in terms of coverage in IPC-2011.

As we discussed before, IPC problem sets are designed to show an incremental difficulty, therefore they are not suitable for our evaluation. In addition, not all domains are interesting for analyzing the search space discrimination. On the one hand, a class of problems in a domain might not be challenging for planners, therefore the runtime for solving these problems is a straightforward function of the problem size, no matter the particular differences of the search trees. On the other hand, available random problem generators might create a low diversity problem set given the same input parameters, which would lead to similar planner performance, and therefore uninteresting for our study.

For the evaluation we have initially considered the domains from IPC-2011 and IPC-2014 for which we have found random problem generators. The first step we have performed towards the selection of interesting problem sets is the generation of a set of 30 random problems with the same input parameters for each domain. The problem size, determined by these input parameters, was manually adjusted to have non trivial problem that, in most cases, can be solved within 1800 time bound. The specific parameters for each domain will be described in the *Results* section. Planners were run on these problem sets to visualize the performance profile of each domain.

Table 1 shows the coefficient of variation (i.e., the ratio of the standard deviation to the mean, $C_v = \sigma/\mu$) for the runtime each planner used in solved instances. With this relative measure of dispersion one can compare performance in different domains and realize which of them have more variety in their performance profile. We have marked in bold the top 10 C_v, which have a value higher than 1. We have focused on these planner/domain performance for the rest of the evaluation. The selected problem sets are from Barman, Depots, Elevators, Floortile, Satellite and TPP.

For each selected problem set, we have generated 170 additional problems to complete a set of 200 training problems. Each planner of interest has been run on these problems to collect the performance data. The complete evaluation was run on a cluster nodes that have an Intel XEON2 2.93Ghz processor, 8 GB of memory and are running on Linux Ubuntu 14.04 LTS.

The Learning Process

For the study we have focused on recognizing if a given planning task is “easy” or “difficult” for a planner. However, from a experimental point of view such concepts are

Domain	Mercury	Probe	Lama
barman	**2.86**	**2.02**	**2.63**
blocksworld	0.53	0.14	0.76
depots	**2.97**	0.08	**1.99**
elevators	0.04	0.22	**1.03**
floortile	0.95	**1.75**	0.68
gripper	0.72	0.04	0.65
nomystery	0.25	-	0.41
parking	0.20	0.69	0.26
rovers	0.49	0.17	0.42
satellite	**1.25**	0.39	0.30
spanner	0.04	0.21	0.08
tpp	**1.57**	0.17	**1.07**
transport	0.02	0.00	0.01

Table 1: Coefficient of variation ($C_v = \sigma/\mu$) of the planner runtime on solved instances. Each set contains 30 homogeneous problems. The top higher values are shown in bold.

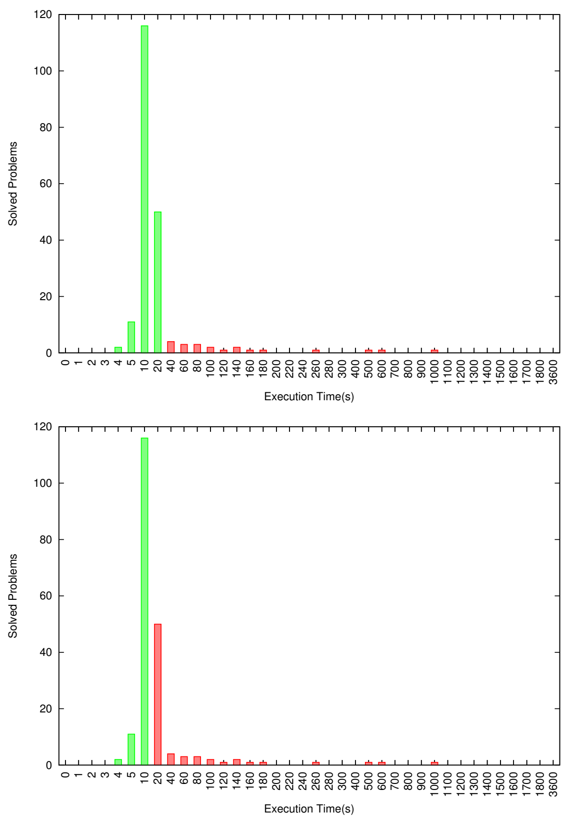

Figure 2: Histogram of Mercury execution runtime for 200 problems of Barman. Green bars represent easy tasks up to percentile (top) 90 and (bottom) 75. Red bars represents hard tasks.

somehow artificial in planning evaluations, since the concept of difficulty changes while planning research advances, and more important it is relative to the time bound used for evaluation. Regarding the homogeneous problem sets, the important key is to differentiate which problems of the same type are difficult, or at least more difficult than the rest. Therefore we say that a problem is difficult for a planner with respect to a (homogeneous) problem set if the runtime to solve such problem is over a given percentile of the time required to solve any problem of the set. Consider for instance Figure 2, where we show histograms of Mercury runtime for the 200 problems of the Barman domain. With the same profile one can use different cut-off points to define the class proportion of easy/hard problems. As we can see, execution time could be said to follow a gamma distribution shape with a long right tail. Such tail could be considered to include the difficult problems if, for instance, we assume percentiles of 90 (top image) or 75 (bottom image). For the evaluation we have used the percentiles 95, 90, 75, and 66 as cut-off points to generate training sets with different easy/hard proportion of problems.

To evaluate the performance of the classifiers, we avoid using classical classification metrics as accuracy, since the data we use might have a unbalanced classes, and different cut-off points could not be compared. Instead, we use the Area Under the ROC Curve (AUROC) metric (Bouckaert 2006). A ROC curve of a binary classifier is a graph where the x-axis represents the false positive rate and the y-axis the true positive rate. The classifier is assumed to be instantiated with a parameter t such that it assigns to a new instance a positive class if the predicted probability to belong to such class is larger than t. When t ranges from 0 to 1, the ROC curve is generated, describing the balance between the true positive and the false positives. For $t = 0$, both true and false positive rates are also 0, since the classifier never outputs a positive value. When $t = 1$ both values are 1, since the classifier returns a positive class for all the instances. The AUROC metric is interpreted as the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. Thus, a random classifier has an AUROC of 0.5, no matter the class balance, given that true and false positive ratios are independent of the number of positive and negative instances. In our experiments, if the computed AUROC for a classifier is larger than 0.5, we can claim that it behaves better than random and provides useful knowledge to classify tasks at the given cut-off point.

For each problem set and cut-off point we trained classification models using 4 machine learning techniques: naive Bayes and decision trees, to have traditional and comprehensive models; and random forest and rotation forest, that have been shown as good learners in recent work on EPMs for planning (Fawcett et al. 2014; Cenamor, de la Rosa, and Fernández 2016). These classifiers were trained using Weka (version 3.6.10), a widely used machine learning Toolkit (Witten and Frank 2005).

To find the features that are able to discriminate easy and hard problems, different feature selection processes could be followed. In this work, we have used a ranker based on the information gain ratio, also provided by Weka. The information gain ratio is a standard discrimination criteria used in the induction of tree-based classifiers, such as decision trees

Planner	Solved	f-set	Real	95%	90%	75%	66%
MERCURY	100	all	-	0.68	0.71	0.71	0.71
		ib2	-	0.68	0.69	0.69	0.69
PROBE	86.5	all	0.72	-	-	0.61	0.63
		ib2	0.67	-	-	0.51	0.60
LAMA	97.5	all	0.46	0.58	0.70	0.70	0.71
		ib2	0.44	0.56	0.69	0.69	0.69

Table 2: AUROC for different difficulty cut-offs in the problem set of Barman. Following Table 1, the three planners are evaluated

or random forest. The feature ranker evaluates the worth of an attribute by measuring the gain ratio with respect to the class.

Performance Modelling

In this section we present the results for the performance modelling on the selected problem sets and the planners of interest. The AUROC results are presented grouped by domains. The results are shown only for the best classifier (i.e., one of the 4 Weka algorithms considered for the study) obtained from a 10-fold cross validation process. The baseline of comparison is the random classifier, which will achieve 0.5 as AUROC by definition. We include the results for:

- **ib2**: models trained using the IBACOP2 feature set (Cenamor, de la Rosa, and Fernández 2016)
- **all**: the previous feature set plus the additional set of features presented in this work

In addition to different cut-off points we include, in the case it makes sense, the results of the "real" proportion of easy/hard problems, computed as the solved/unsolved tasks within the 1800 seconds time-bound. At the end of the section we present the results for the feature rankings and we report an overall analysis with the insights we have found during the process.

Barman

This domain represents a waiter robot that makes cocktails manipulating drink dispensers, glasses and a shaker. The problem set has the following size: 1 shaker, 2 hands, 15 shots, 5 ingredients, 10 cocktails, 5 dispensers. The goals consist of preparing 14 shots. Table 2 shows the AUROC results of the best model for all tested configurations. For LAMA the bad result on the real problem set is expected, given that 5 problems labeled as unsolved is not enough to give at least one instance to test in the 10-fold cross validation scheme. The rest of models show a performance above a random classifier validating the existence of the search space discrimination for these EPMs. The use of the new features slightly increases AUROC values of most models trained with IBACOP2 features. PROBE did not solve 27 of the 200 problem. That is the reason why the cut-offs of 90% and 95% do not make sense.

Depots

This domain is a combination of a transportation domain and the Blocksworld domain. The transportation part involves moving crates around different locations using trucks. Then, at each location, crates are stacked on pallets using hoists. Goals indicate the final position of each crate. The problem set was generated with the following input parameters to the generator: 1 depot, 2 distributors, 2 trucks, 6 pallets, 3 hoists and 20 crates. The results are shown in Table 3. The percentage of solved problem for both LAMA and MERCURY is below 90%, therefore the first two cut-off points do not make sense. For the real class proportion, as well as for rest of cut-off points, the AUROC is very high, achieving the outstanding value of 0.98 for Mercury in the 75% cut-off. The models trained with all features achieved equal or better AUROC than the models using IBACOP2 features.

Planner	Solved	f-set	Real	75%	66%
MERCURY	87.0%	all	0.84	0.98	0.78
		ib2	0.84	0.82	0.78
LAMA	89.5%	all	0.80	0.78	0.75
		ib2	0.71	0.79	0.78

Table 3: AUROC for different profile cut-offs in the problem set of Depots. Following Table 1, only Mercury and Lama are evaluated

Planner	Solved	f-set	Real	95%	90%	75%	66%
LAMA	96.5%	all	0.34	0.53	0.50	0.53	0.54
		ib2	0.34	0.50	0.50	0.53	0.52

Table 4: AUROC for different profile cut-offs in the problem set of Elevators. Following Table 1, only Lama is evaluated

Elevators

This domain represents a set of elevators in a building that move people between different floors. There are slow elevators that stop at each floor within a range of floors and fast elevators that stop every x floors of the building. The goal is to deliver a list of persons to their destination floors. The input parameters of the problem set are: 33 floors, 54 persons, 4 fast elevators stopping at floor numbers multiple of 8, and 8 slow elevators. The AUROC results corresponding to the models trained for LAMA are presented in Table 4. Different from previous domains, these results show a poor prediction performance. For the real proportion, bad results are expected due to the scarcity of negative examples. However, for all cut-off points the performance is still equal or very close to a random classifier. As shown in Table 1, PROBE and specially MERCURY, achieved very low variation on their runtime, and even Lama obtained a specially low value (only 1.03). Those results suggest that these problems are quite similar even in their search space structure.

Floortile

This domain represents a set of robots that paint color patterns on floor tiles. However, robots can only paint the tile that is in front (up) and behind (down) them, and once a tile has been painted a robot can not stand on it. The goal con-

Planner	Solved	f-set	Real	95%	90%	75%	66%
MERCURY	100%	all	-	0.73	0.69	0.77	0.76
		ib2	-	0.68	0.69	0.76	0.74

Table 5: AUROC for different profile cut-offs in the problem set of Satellite. Following Table 1, only Mercury is evaluated

sists of painting a chess-like floor. The problem set was generated with 2 robots on 5x4 grids. The last line is not painted, so robots can stand on it at the end. In this domain we only analyzed the performance of PROBE, following the resuls of Table 1. The problem set was quite challenging for the planner given that it only solved 28% of the problems. Thus, only the problem set with the real cut-off is evaluated. The AUROC for the all-features and ib2-fueatures models were 0.56 and 0.57 respectively. This means that, although the models created are better than a random classifier, it is still difficult to create feasible performance models in this domain. We can not find any further insights from the results given that smaller grids become too easy for all planners, therefore we were unable to set input parameters to the generator to make the problem set interesting.

Satellite

This domain simulates the schedule of observations for a set of satellites. Observation are taken using on-board cameras that support different capture modes. Goals can also include the final pointing direction for each satellite. The set or problems was generated with the following input parameters: 10 satellites with 16 instrument divided among them, 12 capture modes and 122 directions. The only problem set of interest was for the Mercury planner. Results are shown in Table 5. For all cut-off points the AUROC is better than a random classifier. The models trained with all features are equal or better than the models trained only with IBACOP2 features.

TPP

In the *Traveling Purchase Problem* (TPP) domain a set of trucks have to select a subset of markets to go and purchase a set of goods in order to satisfy a given demand that is specified in the goals. The training set was generated with the following size: 16 goods, 6 trucks, 7 depots and 22 markets with 6 levels (i.e., to represent quantity) of goods. Both LAMA and MERCURY solved the 200 instances within 1800 seconds. Table 6 shows the AUROC of the best model for different cut-offs. All models present AUROCs consistently higher than a random classifier. In this domain the benefits of new features is more evident than in the rest of domains given that in all cases AUROC increased in at least 3 percentage points.

Ranking of Features

For each combination of domain/planner we have selected the 75% cut-off problem set to analyze the relevance of features. The gain ratio ranker was executed on a 10-fold partition and then the average rank for each feature was computed. In Table 7 we present a summary of the results, which lists the features that achieved an average rank ≤ 5.0 in at least one performance dataset. Blank values represent a null contribution to the gain ratio and therefore a position with no meaning in the ranking. Indeed, this is a tie with all other features that do not provide useful knowledge. The *Best* column shows the best average rank over all datasets, and *N* column computes the number of times (performance datasets) a feature provides useful knowledge. The features are grouped by category. The Floortile/PROBE combination is not included because there is no performance data for the 75% cut-off. Results for the Elevator/LAMA dataset is not shown either because all features have a null contribution to the gain ratio. This makes sense given the bad results shown in Table 4.

Planner	Solved	f-set	Real	95%	90%	75%	66%
MERCURY	100%	all	-	0.79	0.76	0.78	0.84
		ib2	-	0.75	0.72	0.75	0.77
LAMA	100%	all	-	0.86	0.79	0.76	0.73
		ib2	-	0.81	0.66	0.68	0.69

Table 6: AUROC for different profile cut-offs in the problem set of TPP. Following Table 1, only Mercury and Lama are evaluated

As expected, none of the PDDL features appear in the list. In fact, the only feature that does not have a null contribution to the gain ratio in the number of goals. However its contribution is not significant, as its best average rank was 22.6 in the Depots/LAMA data. Number of goals in Depots and Satellite can have small variations due the list of goals is created by the generators with additional source of randomness. Five features from the Causal Graph appear as relevant, specially in Barman. The first interesting observation is that even these problems are quite similar at instantiation level, there are several features that are different enough to help in the discrimination of hard problems. Five fact balance features also appear as relevant. The new footprint features are more informative than previous ones, as we can see that for instance the *balancedRP* contributes to discrimination in 6 out of 8 possibilities. The best average rank for *DistortedRP* was 10.3 in Depots/LAMA.

The heuristic values of the initial state are also recognized as relevant features. The red-black heuristic gave some information in six times, included all modelling for Mercury, but its contribution was very relevant only for TPP/LAMA. Here we can see that features not necessarily contribute in the performance of planner they are related. The best average rank of the FF heuristic was 8.3 in Lama/TPP. Surprisingly, the goal count heuristic was relevant in Depots. The reason is that random selection of final destination for crates can coincide with their initial state, and therefore problems are partially discriminated by the number of unachieved goals in the initial state. Note that this is different from the number of goals at PDDL level, which makes a syntactical count of goals. Four of the new features from the landmark graph are listed as relevant. Having a average rank of 1.0 from a

			Barman			Depots		Satellite	TPP			
Type	Attribute	New	Mercury	Lama	Probe	Mercury	Lama	Mercury	Mercury	Lama	Best	N
CG & DTG	totalEdgesCG		-	2.9	5.9	-	-	-	-	30.7	2.9	3
	variable/edges ratio		-	4.6	4.5	-	-	16.0	-	22.2	4.5	4
	varWeight/edges ratio		-	6.1	3.3	-	-	-	-	22.0	3.3	3
	inputEdgeCG avg		-	2.1	4.4	-	-	6.7	-	23.9	2.1	4
	outputEdgeCG avg		-	3.9	4.6	-	-	6.9	-	23.6	3.9	4
Fact Balance	fact balance avg		-	-	-	3.7	41.4	-	-	-	3.7	2
	goal balance avg		-	-	-	13.9	3.8	-	-	-	3.8	2
	goal balance var		-	-	-	20.6	3.4	-	-	-	3.4	2
	balancedRP	✓	6.7	-	-	70.1	60.2	73.3	2.5	2.4	2.4	6
	unbalanceRP	✓	20.4	-	-	21.9	-	68.2	4.1	63.1	4.1	5
Heuristic	Additive		7.4	-	-	5.8	59.0	-	9.5	4.8	4.8	5
	Causal graph		12.2	-	-	1.9	3.6	-	11.2	6.5	1.9	5
	Goal count		-	-	-	5.3	2.4	46.2	-	-	2.4	3
	Landmark cut		-	-	-	28.0	66.6	30.7	5.0	8.5	5.0	5
	Red-black	✓	23.9	-	-	29.4	38.7	28.9	17.7	2.7	2.7	6
Landmark Graph	numberEdges	✓	1.0	-	-	10.6	16.5	7.0	4.6	38.8	1.0	6
	landm/edges ratio	✓	4.7	-	-	25.4	-	61.3	11.9	-	4.7	4
	inputEdges max	✓	5.9	-	-	56.7	-	1.0	-	-	1.0	3
	inputEdges std	✓	10	-	70.1	2.4	-	17.2	19.1	-	2.4	5

Table 7: Results of the gain ratio ranker for features that achieved an average rank $\leq$ 5.0 in at least one domain/problem configuration.

10 fold partition is a clear evidence that the number of edges and maximum number of incoming edges of the landmark graph are relevant features for the space search discrimination when modelling Mercury's performance. After an overall review of this result we can conclude that features are more or less relevant depending on the planner and the problem set. For this reason we think that there is room for including additional features that can improve the overall prediction capabilities on homogeneous problem sets.

Conclusions and Future Work

The main objective of this work was to verify whether known features for characterizing planning tasks are able to encode knowledge for the classification of hard tasks in scenarios where performance models have to discriminate between problems of the same input configuration. Results have shown that considering the ten most diverse performance data from a set of planners and domains, the performance models systematically behave better than random classifiers. In addition, we have extended the IBACOP2 feature set and verified empirically that the new features improved the prediction power of performance models. Most of EPMs trained on homogeneous problem sets are still far from perfect classifiers, so we think there is room for aggregating additional features that characterize other aspects regarding the search space discrimination. As we saw, the relevance of the features is not dominant across different domains and planners, therefore new features could add robustness to EPMs.

As future work we want to include other features extracted from search probes, without compromising too much the computational cost of computing them. Moreover, we will make the problem sets generated for the study available to the public for further analysis and comparisons. We also have the interest of developing random generators of hard problems. The idea consists of making a wrapper of the available problem generators and produce only the instances that are considered above certain cut-off of difficulty as provided by the corresponding model prediction.

Acknowledgments. This work has been partially supported by the Spanish projects TIN2014-55637-C2-1-R and TIN2015-65686-C5-1-R.

References

Bouckaert, R. R. 2006. Efficient auc learning curve calculation. In *Australasian Joint Conference on Artificial Intelligence*, 181–191. Springer.

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2012. Mining ipc-2011 results. In *Proceedings of the Third Workshop on the International Planning Competition.*

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2013. Learning predictive models to configure planning portfolios. In *Proceedings of the Workshop on the Planning and Learning*.

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2016. The IBaCoP planning system: Instance-based configured portfolios. *Journal of Artificial Intelligence Research (JAIR)* 56:657–691.

Domshlak, C.; Hoffmann, J.; and Katz, M. 2015. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence* 221:73–114.

Fawcett, C.; Vallati, M.; Hutter, F.; Hoffmann, J.; Hoos, H. H.; and Leyton-Brown, K. 2014. Improved features for runtime prediction of domain-independent planners. In *In Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS-14).*

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26(1):191–246.

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

Hoffmann, J. 2011. Analyzing search topology without running any search: On the connection between causal graphs and h+. *Journal of Artificial Intelligence Research* 41:155–229.

Lipovetzky, N., and Geffner, H. 2011. Searching for plans with carefully designed probes. In *In Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS-11)*, 154–161.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 975–982.

Richter, S.; Westphal, M.; and Helmert, M. 2011. Lama 2008 and 2011. *The 2011 International Planning Competition* 50.

Roberts, M., and Howe, A. 2009. Learning from planner performance. *Artificial Intelligence* 173(5):536–561.

Roberts, M.; Howe, A. E.; Wilson, B.; and desJardins, M. 2008. What makes planners predictable?. In *ICAPS*, 288–295.

Vidal, V. 2004. A lookahead strategy for heuristic search planning. In *In Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, 150–160.

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

Zhu, L., and Givan, R. 2003. Landmark extraction via planning graph propagation. In *ICAPS Doctoral Consortium*, 156–160.