Journal of Artificial Intelligence Research 56 (2016) 657-691 Submitted 01/16; published 08/16

Engineering Note

The IBaCoP Planning System: Instance-Based Configured Portfolios

Isabel Cenamor ICENAMOR@INF.UC3M.ES
Tomás de la Rosa TROSA@INF.UC3M.ES
Fernando Fernández FFERNAND@INF.UC3M.ES
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. Leganés (Madrid). Spain

Abstract

Sequential planning portfolios are very powerful in exploiting the complementary strength of different automated planners. The main challenge of a portfolio planner is to define which base planners to run, to assign the running time for each planner and to decide in what order they should be carried out to optimize a planning metric. Portfolio configurations are usually derived empirically from training benchmarks and remain fixed for an evaluation phase. In this work, we create a per-instance configurable portfolio, which is able to adapt itself to every planning task. The proposed system pre-selects a group of candidate planners using a Pareto-dominance filtering approach and then it decides which planners to include and the time assigned according to predictive models. These models estimate whether a base planner will be able to solve the given problem and, if so, how long it will take. We define different portfolio strategies to combine the knowledge generated by the models. The experimental evaluation shows that the resulting portfolios provide an improvement when compared with non-informed strategies. One of the proposed portfolios was the winner of the Sequential Satisficing Track of the International Planning Competition held in 2014.

1. Introduction

Planning is a process that chooses and organizes actions by anticipating their outcomes with the aim of achieving some pre-stated objectives. In Artificial Intelligence, Automated Planning (AP) is the computational study of this deliberation process (Ghallab, Nau, & Traverso, 2004). Automated planners are systems that, regardless of the application domain, are able to receive a declarative representation of an environment, an initial state and a set of goals as input. The output is a synthesized plan that will achieve these goals from the initial situation. In this context, the International Planning Competition (IPC) is an excellent initiative to foster the studying and development of automated planning systems. IPC was created in 1998 to set a common framework for comparing automated planners.

Different planning systems won awards in previous IPCs. However, one of the main invariants of the competition is that there is no single planner which is always the best planner (or at least equal) for every domain or every problem. This means that, although there is a planner which, following the quality metrics of the competition, can be considered the best, we can always find some problems in different domains in which other planners outperform the overall winner. Therefore, we can assume that the AP community has generated a set of single planners that are better than all others in specific situations. For this reason, discarding "a priori" any of those solvers seems meaningless.

In fact, the idea of reusing a set of individual or base systems to generate more accurate solutions than those obtained separately is not new in Artificial Intelligence. For instance, in Machine Learning, meta-classifiers use different base classifier to increase the coverage of the representation bias of the resulting classifier (Dietterich, 2000). In problem solving, portfolios of search algorithms have also demonstrated that they can outperform the results of a single search strategy (Xu, Hutter, Hoos, & Leyton-Brown, 2008; Xu, Hoos, & Leyton-Brown, 2010; Malitsky, Sabharwal, Samulowitz, & Sellmann, 2013). For example, the SAT competition in 2013 included a special track on portfolios. In the automated planning community, planner portfolios have also been subject to great deal of interest. In IPCs from 2006 to 2014, portfolio approaches won or were very close to winning the tracks in which they took part.

However, although the use of portfolios has become usual in the community, there is still no agreement as to what a planning portfolio is (Vallati, Chrpa, & Kitchin, 2015). In this work, we assume that a portfolio of planners is a set of base planners with a selection strategy. This selection strategy is what generates a specific portfolio configuration, whose goal is to maximize the performance metrics. Therefore, a configuration has to define three main elements: (1) **which sub-set of planners to run**, (2) **how long to run each planner?** and (3) **in which order**. There are many techniques to configure a planning portfolio (Vallati, 2012), and depending on how accurate they are, the chances of selecting the best planner in a given situation will increase. Note that, in this definition, if a planner has different configuration parameters which modify its behavior, each parameterization is considered a different base planner, so base planners can be considered as black boxes.

The number of planners in the state of the art is huge, so a first filtering is to select the minimum number that ensures the best performance is achieved, for each evaluated planning domain (or even for each problem in each domain). Obviously, good results in current domains do not ensure good results in new domains but, as will be shown, it is a good estimator. In this sense, a Pareto efficiency-based approach (Censor, 1977) to reduce the number of planners that we consider eligible for a planning portfolio is presented. However, we will show that with this mechanism, the first of the aforementioned questions can only be answered partially since the number of candidate planners might still be large.

So the best solution to the portfolio configuration problem is to have an oracle that predicts, given a domain and a problem, which planner will obtain the best performance and how long it will take. Given that we do not have this oracle, in this work we propose the use of predictive models, automatically generated with Machine Learning and Data Mining techniques. These models summarize the results of all the candidate planners from the past: whether they were able to solve planning problems, as well as the time that they required to generate a good solution (Cenamor, de la Rosa, & Fernández, 2012, 2013). Given this knowledge on the past, the inductive hypothesis gives also us an estimation on how they will behave in future planning domains and with different problems, so the order in which the planners are implemented can be given by the accuracy of these predictions. Therefore, with these predictive models, we are able to configure a portfolio for each planning problem, like in previous works on the use of portfolios in search (Gomes & Selman, 2001). This is a renewed idea in automated planning since recent works have focused in static (Helmert, 2006) or domain-specific portfolios (Gerevini, Saetti, & Vallati, 2009, 2014), in which the configuration of the portfolio is fixed for all the domains or chosen for each one respectively.

IBACOP (*Instance-based Configured Portfolio*) is a family of planning portfolios that were built for competing in IPC-2014. In this article we first present IBACOP as a general framework with the ultimate goal of building per-instance configurable portfolios. The technique can be reproduced again whenever new automated planners or new planning benchmarks arise. Then, we describe how to build different version of IBACOP following the defined processes. One of these versions was the winner of the Sequential Satisficing Track of IPC-2014. We also include the results of an empirical study that confirms the good performance of IBACOP planners when compared to different base planners and different portfolio configuration strategies. Then, we summarize the related work, and finally, the last section sets out the conclusions and future lines of research.

2. System Architecture

In this section, we present the general idea of building a planning portfolio that can be configured for a particular planning task using predictive models. This process should be seen as a general technique given that the inputs (planners and benchmarks) might change in the future due to progress in the planning community, so new portfolio configurations can be generated through the use of these new inputs.

2.1 Portfolio Construction

We consider that the construction of an instance-based planning portfolio comprises three main parts. (1) Planner filtering, for making a pre-selection of good candidate planners from the set of known or available planners. The proposed pre-selection technique is based on a multi-criteria approximation. This is a previously unexplored technique for selecting a set of planners that provides enough diversity in the planner portfolio. (2) Performance modeling, for providing predictors of the planner's behavior as a function of planning task features. In our research, we include a set of well-known features (Cenamor et al., 2012), some of which are built into the preprocessing step of FAST DOWNWARD (Helmert, 2006). We also take advantage of both the output information in the translation process (Fawcett, Vallati, Hutter, Hoffmann, Hoos, & Leyton-Brown, 2014) and the heuristic values computed in the first step of the search process of FAST DOWNWARD. In addition, the use of several totally new features on the characteristics of the relaxed plan in the initial state is proposed. Finally, (3) strategy selection: to establish a procedure that combines the performance predictions and then to output a portfolio configuration. We propose a novel strategy selection to exploit the effectiveness of the predictive models. Next, we explain the details of each of these construction steps.

2.1.1 PLANNER FILTERING

The planner filtering process consists of the pre-selection of good candidate base planners from a larger amount of available planners. Even though there is a sufficient evidence that there is not an overall best planner across a variety of benchmarks, it can be verified empirically that there is a dominance of some planners over others. Therefore it does not make sense to include, as base planners, those that are always worse in terms of performance metrics. We want this filtering process to select a diverse, but small, subset of planners to have few elements among which to divide the available execution time.

In this work, we propose a multi-criteria pre-selection mechanism that focuses in two IPC metrics (quality and time) as alternative to the most extended ones for planner filtering. For example, FDSS (Helmert, Röger, Seipp, Karpas, Hoffmann, Keyder, Nissim, Richter, & Westphal, 2011) uses the selection of planners that maximizes the coverage; MIPLAN (Núñez, Borrajo, & Linares López, 2015) uses the portfolio configuration that obtains the best achievable performance in terms of score.

For filtering we propose to run the candidate planners on a representative set of benchmarks and then to evaluate them in terms of time and quality. To consider both metrics we propose an approach based on Pareto-efficiency (Censor, 1977) that allows us to determine the dominance between planners in a multi-criteria fashion. In particular, we select a planner as a candidate for the portfolio if it is the best planner for at least one domain in terms of the IPC-2011 multi-criteria QT score (Linares López, Celorrio, & Olaya, 2015). Briefly, for a single problem, this metric computes the tuple $\langle Q, T \rangle$ for each planner, where Q is the quality of the planner's best solution and T the time used to find this solution. Then, for a given planner, p, the dominance relations between p and the rest of planners are computed.

A tuple $\langle Q, T \rangle$ Pareto-dominates the tuple $\langle Q', T' \rangle$ if and only if $Q \geq Q'$ and $T < T'$. Planner p gets $\frac{N}{N^*}$ points, where N is the number of tuples where p Pareto-dominates another planner, and N^* is the number of different tuples in which planner p appears. Finally, the QT-Pareto score for a domain is a sum of points achieved in all the problems in the domain. The idea of this selection mechanism is as follows: if a planner shows good dominance property in a given domain, it should be included in the portfolio because it will be a good candidate for solving the problems of the same domain or even other planning tasks that have similar characteristics. Therefore, a simple strategy to filter a first pool of planners is given by the procedure that selects only the planners with the maximum QT-Pareto score for at least one domain. We refer to this procedure as *QT-Pareto Score Filtering*.

2.1.2 PERFORMANCE MODELING

Given a planning task, we want to predict how the selected base planners will perform in order to decide whether to include them or not and to make a good assignment of time and ordering when configuring the portfolio. Thus, modeling the planner behavior as a function of planning task features becomes a key process in building instance-based portfolios. To learn these predictive models we follow a Data Mining approach, as shown in Figure 1. In our case, we start from a set of candidate planners and a set of planning benchmarks. The output of the process is the set of models that will predict the performance of the candidate planners. We have defined the data mining goal as the creation of two predictive models. First, whether a planner will be able to solve a problem (i.e. a classification task) and, if so, what will be the time required to compute the best plan (i.e., a regression task).

The first step of the mining process comprises the generation of training and test datasets. On the one hand, the planners are run on the set of benchmarks to obtain their performance data. This data includes the outcome of the execution (success or failure) and, for the positive cases, the time elapsed in finding the best solution. On the other hand, planning tasks are processed to extract a set of features that characterize them. These features are an extended set of the previously proposed set (Cenamor et al., 2013). According to the mechanism for generating these features, we classify them into the following categories:

- PDDL features: Basic features extracted from the PDDL representation of the domain and problem files, for instance, the number of actions, objects or goals.

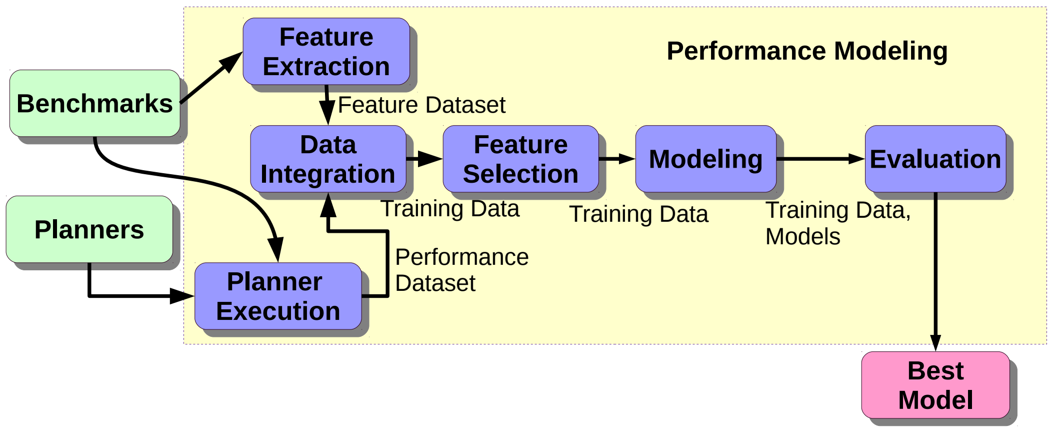

Figure 1: General Diagram for Learning the Planning Performance Predictive Models

- FD Instantiation features: the Fast-Downward pre-processor instantiates and translates the planning tasks into a finite domain representation (Helmert, 2009). From this output we take some general information such as the number of instantiated actions or the number of relevant facts, and data specific to the FD-translator, such as the number of auxiliary atoms.

- SAS$^+$ features: The finite domain representation of SAS$^+$ has an associated *Causal Graph* (CG) and a set of *Domain Transition Graphs* (DTGs). From CG we extract basic properties (e.g., number of variables and edges), and the ratios between these properties. As regards DTGs, the number of graphs in a problem corresponds to the number of edges in the CG, which makes it difficult to encode the general attributes for each DTG. Therefore, we summarize the DTGs characteristics by aggregating the relevant properties of all graphs. Thus, features from DTGs are statistics on them such as the maximum, the average or the standard deviation of their graph properties.

- Heuristic features: For the initial state, we compute heuristic values using a set of widely-used unit cost heuristic functions (e.g., h_{max}, h_{FF},...). We compute these heuristics only for the initial state, which can be obtained at a reasonable cost. We use only unit cost heuristics to obtain a domain-independent estimation that helps in the characterization of the problem size and/or difficulty.

- Fact Balance Features: Using the relaxed plan (RP) of the initial state, extracted when computing the h_{FF} heuristic, we also compute a set of features to represent the *fact balance* of the RP. We define the fact balance for fact p, as the number of times that p appears as an added effect of an action belonging to RP, minus the number of times that p is a deleted effect of an action in RP, considering original actions where deletes are not ignored. The intuition behind fact balances is that high positive values would characterize easier (relaxed) problems for a given domain, since achieved facts do not need to be deleted many times. Given that the number of relevant facts of a planning task is variable, we compute statistics (i.e., min, max, average and variance) for the fact balance of the relevant facts. Additionally, we compute statistics only by considering facts that are goals, following the same procedure.

The complete set of 89 features is listed and organized by their category in Appendix A. The *Data Integration* process in Figure 1 receives the features and the performance datasets as inputs to produce a final dataset according to the modeling goal. In the dataset for the classification task, a training/test instance includes the planning task features plus the planner name and the Boolean feature indicating whether this planner solved the planning task. The dataset for the regression task only includes the cases in which the planning tasks are solved. We make this exclusion because it does not make sense to model or estimate the planning time beyond the given time limit and because in most cases this time is unknown. A training/test instance in the regression dataset includes the planning task features, the planner name and the time this planner used to find its best solution.

The *Feature Selection* is an optional process for reducing the number of features used for the modeling. This procedure is applied because there might be irrelevant or redundant features that could degrade the modeling capabilities of some learning techniques (Blum & Langley, 1997). The outcome of the process is dependent on the original data. Thus, the decision of whether to apply it or not is taken based on the results of the model evaluation.

For the *Modeling* process, we use an off-the-shelf data-mining tool that provides a set of learning algorithms for both classification and regression. The generated models are then evaluated in the *Evaluation* process to determine the best model for the classification and regression tasks. There are many different ways of carrying out the model evaluation and comparison (Han, Kamber, & Pei, 2011; Witten & Frank, 2005), which will reflect the generalization ability of the different models when making predictions of unseen data.

2.1.3 STRATEGY SELECTION

The strategy selection is the final step in the construction of an IBACOP planner. Selecting a strategy implies that we have to decide how to transform the predictions of the best models into an actual portfolio configuration. There are several alternatives that range from ignoring both model predictions to trusting them completely. For the classification model, each candidate planner will get a yes/no prediction given a new planning task. The direct use of the Boolean variable makes difficult to decide which planners to include in the portfolio. Consider, for instance. the two extreme cases: (1) If all planners get a positive prediction, should we include all of them? (2) If all planners get a negative prediction, which planner should we include in the portfolio? Instead of using the Boolean prediction we propose to rank the predictions by their confidence in the positive class, and then make the selection of planners according to this ranking. Then, each planner should be assigned a slide of the total time, in which this assignment can be carried out uniformly or dependently, again, from the predictive models learned. Therefore, depending on the use that we make of the predictive models, we propose three basic strategies:

1. *Equal Time for all (ET)*: This strategy does not use the predictive models at all. It will assign equal time for each planner (uniform strategy). The idea behind this strategy is to have more planners but with less time for each one. This strategy has obtained good results in other portfolios (Seipp, Braun, Garimort, & Helmert, 2012).

2. *Best N confidence (BN)*: This strategy will include the subset of N planners with the best prediction confidence in the positive class in the portfolio. Then, they get equal time for solving the planning task. In this case, the idea is that we select a subset of promising planners so they can spend more time in solving the planning task.

3. *Best N Estimated Time (BNE)*: The subset of planners is selected as mentioned before, but now the time is assigned proportionally to the estimated time provided by the regression model.

2.2 Portfolio Configuration

An instance-based configuration of a portfolio implies that the subset of base planners and the time assigned to each one varies as a function of the planning task features. The set of candidate planners, the predictive models and the configuration strategy are previously fixed in the construction phase. Algorithm 1 shows how to use these components to configure the portfolio for a given planning task.

Algorithm 1: Algorithm for configuring the portfolio for a particular planning task.

```
Data: Problem ($\pi$), Domain ($d$), Set of base planners ($\mathcal{P}_{ini}$), Classification model ($\mathcal{C}$),
      Regression model ($\mathcal{R}$), Available time ($T$), Strategy ($\mathcal{S}_N$)
Result: Portfolio Configuration: A sequence of planners with their assigned runtime,
        $Portfolio = [\langle p_1, t_1\rangle, \ldots, \langle p_c, t_c\rangle]$
Portfolio=[];
if $\mathcal{S}_N$ == ET then
    /*(No classification nor regression models available)*/
    $n = size(\mathcal{P}_{ini})$;
    for $p\ in\ \mathcal{P}_{ini}$ do
        $append(\langle p, \frac{T}{n}\rangle$, Portfolio);
else
    $\langle F, t_F\rangle = extractFeatures(d, \pi)$;
    for $p_k\ in\ \mathcal{P}_{ini}$ do
        prediction$\langle p_k, conf_k^{\oplus}\rangle \longleftarrow$ predict $(\mathcal{C}, \langle F, p_k\rangle)$;
    sorted_candidates $\longleftarrow$ sort(prediction, $key = conf^{\oplus}$);
    $p' \longleftarrow$ sorted_candidates$[i \ldots N]$;
    if $\mathcal{S}_N$ == BN then
        /*Classification model available, applying Best $N$ confidence strategy*/
        for $i = 1\ to\ N$ do
            $append(\langle p'_i, \frac{T-t_F}{N}\rangle$, Portfolio);
    else
        /*Regression model available, applying Best N Estimated Time*/
        for $i = 1\ to\ N$ do
            $t_i = predict\_time(\mathcal{R}, \langle F, p'_i\rangle)$;
        $t' = scaleTime(t, T - t_F)$;
        for $i = 1\ to\ N$ do
            $append(\langle p'_i, t'_i\rangle$, Portfolio) ;
```

The method receives a problem (π), a domain (d), the set of base planners ($\mathcal{P}_{ini}$), the classification model ($\mathcal{C}$), the regression model ($\mathcal{R}$), the time available (T) and the portfolio configuration strategy ($\mathcal{S}_N \in \{ET, BN, BNE\}$). The procedure calls several functions described below:

- *extractFeatures*: This is the same feature extraction procedure used in the portfolio construction phase. From the pair (domain, problem) the function outputs the set of features F. This function also computes the time (t_F) as the time spent in extracting all features.
- *predict*: This function is a query to the classification model $\mathcal{C}$. It receives a new instance represented by the tuple $\langle F, p\rangle$, where F is the previously computed features, and p is the planner name. From the result of the function we ignore the class, and only keep the prediction confidence of the positive class, forming the tuple $\langle p, conf^{\oplus}\rangle$. This output represents the confidence that the planner p will solve the problem.
- *predict_time*: This function uses the model $\mathcal{R}$ to estimate the execution time for the subset of planners $\mathcal{P}_N \subseteq \mathcal{P}_{ini}$ that has been established as the best N candidates in terms of classification confidence. As in the classification model, this function receives the input tuple $\langle F, p\rangle$.
- *scaleTime*: This function transforms the vector of estimated times into another proportional vector for which its sum fits in the available time, which is the original time bound T minus the time used to compute the features t_F. Thus, the time t' assigned to each planner is computed with the formula $t' = \frac{(T-t_F)*t}{\sum_{i=1}^{N} t_i}$

The output of the algorithm is a sequence of planners and their assigned time. The execution of a particular configuration of the portfolio comprises the sequential execution of these base planners ensuring that each CPU process does not exceed the assigned time.

3. IBaCoP Planning System

In this section we describe how we follow the approach presented in Section 2 to build different portfolios.

3.1 Candidate Planners

The initial set of planners includes the 27 planners of the Sequential Satisficing Track of IPC-2011 plus LPG-TD (Gerevini, Saetti, & Serina, 2006). Although LGP-Td did not compete in IPC-2011 we considered worthwhile to include it because it is still considered a state-of-the-art planner due to its great performance in previous competitions.

The first step is to apply the QT-Pareto Score Filtering described in subsection 2.1.1 to reduce the initial set of candidate planners. The benchmarks for computing the QT-Pareto Score is the set of domains and problems of the Sequential Satisficing Track of IPC-2011.

Table 1 shows the best planner in terms of QT-Pareto score for each domain. Additionally, we include the number of problems solved by the best planner to highlight the correlation among both values. The QT-Pareto score values closer to 20 reflect that the planner is able to beat the other planners in most problems. PROBE was the best planner in 4 domains. However the other planners only stood out in one domain. This reinforces the motivation to find a diverse subset of planners. Finally, out of 28 initial planners, the QT-Pareto score filtering pre-selected as candidate planners the subset of 11 planners, which was made up of: LAMA-2011, PROBE, ARVAND, FDSS-2, FD-AUTOTUNE-1, FD-AUTOTUNE-2, LAMAR, LAMA-2008, MADAGASCAR, YAHSP2-MT and LPG-TD. A brief description of these planners can be found in Appendix D.

Planner	Domain	QT	Coverage
PROBE	scanalyzer	16.59	20
PROBE	woodworking	18.55	20
PROBE	tidybot	16.77	18
PROBE	barman	19.42	20
ARVAND	pegsol	18.88	20
MADAGASCAR	parcprinter	17.63	20
LAMA-2008	transport	17.84	19
LAMA-2011	openstacks	17.30	20
FD-AUTOTUNE-1	sokoban	17.56	19
FD-AUTOTUNE-2	nomystery	16.73	19
FDSS-2	elevators	17.84	20
LAMAR	parking	18.12	20
YAHSP2-MT	visitall	18.74	20
LPG-TD	floortile	11.96	12
total		243.77	267

Table 1: List of the best planners ordered by their QT-Pareto score for each domain of IPC-2011.

Table 2 shows the ranking of planners of the IPC results (i.e., planner ordering established by the quality score) (Linares López et al., 2015) and which of them were selected by QT-Pareto Score Filtering. It is worth noting of attention that 10 of the 11 best planners in the IPC are built on top of FD, which reduces the diversity of the planners. However, the QT-Pareto Score Filtering only includes 8 of them. In addition, it should be pointed out that the last three selections of the QT-Pareto Score Filtering are planners from the lower positions of the table which, as will be demonstrated later, increases the diversity of the portfolio and its performance.

Ranking	planner	Eligible	FD
1	LAMA-2011	$\surd$	$\surd$
2	FDSS-1		$\surd$
3	FDSS-2	$\surd$	$\surd$
4	FD-AUTOTUNE-1	$\surd$	$\surd$
5	ROAMER		$\surd$
6	FORKUNIFORM		$\surd$
7	FD-AUTOTUNE-2	$\surd$	$\surd$
8	PROBE	$\surd$	
9	ARVAND	$\surd$	$\surd$
10	LAMA-2008	$\surd$	$\surd$
11	LAMAR	$\surd$	$\surd$
17	YAHSP2-MT	$\surd$	
22	MADAGASCAR	$\surd$	
24	LPG-TD	$\surd$	

Table 2: List of 11 best planners ordered by its score at IPC-2011. The third column shows whether they are selected by the QT-Pareto Score Filtering. The forth column shows if the planners are built on the top of FD.

3.2 Performance Models

The inputs to the performance modeling phase are the candidate planners (i.e., the 11 candidates selected in the previous section) and the benchmark planning tasks selected for this purpose. Next, we describe the generated training data, and then how these inputs produce specific instances of IBACOP planners.

3.2.1 TRAINING DATA

The training data for the learning process requires a set of domains and problems used to gather the input features. We need a wide range of domains and problems to generalize future unknown planning tasks properly. We have included the planning problems available from IPC-2006 onwards. If we do not mention the test set explicitly, it will always refer to the satisficing tracks of the competitions. The included domain and problems are:

- IPC-2006: openstacks, pathways, rovers, storage, tpp and trucks.
- IPC-2008: cybersec, elevators, openstacks, pegsol, pipesworld, scanalyzer, sokoban, transport and woodworking.
- IPC-2011: barman, elevators, floortile, nomystery, visitall, tidybot, openstacks, parcprinter, parking, pegsol, sokoban, scanalyzer, transport and woodworking.
- Learning track IPC-2008: gold-miner, matching-bw, n-puzzle, parking, thoughful and sokoban.
- Learning track IPC-2011: barman, blockworld, depots, gripper, parking, rovers satellite, spanner and tpp.

From this list we obtained 45 different domain descriptions. Although some of them represent alternative encodings of the same domain, all have been included. Candidate planners were run on these benchmarks to obtain the features related to the performance of the planners. Thus, we used a total of $1,251$ planning tasks. The performance data comprises $13,761$ instances (i.e., $1,251$ problems $\times$ 11 planners) where $8,697$ were successful and $5,394$ failed. The proportion of instances solved by each candidate planner is different. Table 16 in Appendix C shows a per-planner summary of the performance data.

The 89 features representing each planning task are automatically generated from the domain and problem definitions. The PDDL features, FD instantiation and SAS$^+$ features are computed using the FAST-DOWNWARD pre-processor. The computation time needed to extract these features is negligible compared to the SAS$^+$ translation, given that we only compute sums and statistics on the data provided by the SAS$^+$ representation. The heuristic features are computed using the FAST-DOWNWARD search engine, and fact balance features are generated using the relaxed planning graph structures (of the initial state) provided by the FF planner (Hoffmann, 2003). The FAST-DOWNWARD pre-processor could fail when instantiating a planning task. In which case, regarding features are not computed and missing values are assumed.

Table 3 shows the success rate for extracting the features of each type from the training problems, and the average and maximum time in seconds to extract them. The PDDL, FD and SAS$^+$ features are extracted from the FD pre-processor which is why they have the same success rate. The time required to compute the heuristic features is only the time for calculating the heuristic value of the initial state, which is calculated only if the FD pre-process has finished successfully.

Class	**Success**	**Average (s.)**	**Max (s.)**	**# features**
PDDL	97%	6.97	46.00	8
FD	97%	52.73	141.40	16
SAS^+	97%	22.60	60.60	50
Heuristic	87.54%	20.20	30.50	8
Fact Balance	93%	5.20	21.20	7
Total	-	107.7	299.7	89

Table 3: Summary of the extracted features with the average and maximum time in seconds (s.) to extract them. These processes are on the top of the two first step of the all planners based on FD.

3.2.2 Feature Selection

We have carried out a feature selection process for two main reasons. On the one hand, some features might be irrelevant whilst others might be redundant for the modeling purpose. Therefore we want to analyze whether it is possible to obtain better models using only a subset of the available features. On the other hand, this study will allow us to recognize most relevant features for characterizing a planning task.

The feature selection was carried out using J48 algorithm, a top-down induction algorithm to build decision trees (Quinlan, 1993), by selecting the features that appear in the top nodes of the tree (Grabczewski & Jankowski, 2005). Decision trees make an implicit feature selection as the model includes queries to those features considered relevant. After applying this feature selection process on the feature dataset, the total number of features decreased from 89 to 34. This leads to a dataset size reduction of around 62%. Table 4 contains the list of features resulting from the feature selection process. The selection chooses features from all categories. For the modeling and evaluation process we kept both datasets separate, one with all available features (f-all) and the other one with the selected features (f-34).

3.2.3 Classification Models

We have trained the classifiers using 31 classification algorithms provided by Weka (Witten & Frank, 2005), which includes different model types such as decision trees, rules, support vector machines and instance based learning. We recall that training instances include the planning task features described in Section 2.1.2 plus the planner name and the Boolean feature indicating whether this planner solved the planning task or not. The performance of the predictive models was evaluated with a 10-fold cross-validation on a uniform random permutation of all training data. The best model for both datasets f-all and f-34 was that generated by Rotation Forest (Rodriguez, Kuncheva, & Alonso, 2006), achieving 93.39 and 92.35% of accuracy respectively. These results are quite better than the result of the default model (ZeroR), which obtained 61.72% of accuracy. See all the results of the classification models in Table 14 of Appendix B.

Even though a good accuracy in the classification model does not guarantee a good performance of the portfolio, this result is a great starting point for selecting promising planners. The accuracy results of the feature selection only showed small differences compared to results obtained with all

Type	Features	Type	Features
PDDL (4)	types	**FD** (6)	auxiliary_atoms
	goal		implied_effects_removed
	objects		translator_facts
	functions		translator_total_mutex_groups_size
			num_relevant_facts
			num_instance_actions
CG & DTG (11)	numberVariablesCG	**Heuristics** (7)	Additive
	inputEdgeCGStd		Context-enhanced_additive
	outputEdgeCGAvg		FF
	outputWeightCGMax		Goal_count
	outputWeightCGAvg		Landmark_count
	outputEdgeHVStd		Landmark-cut
	outputWeightHVMax		Max
	numberVariablesDTG	**Balance** (6)	rp_fact_balance_avg
	totalEdgesDTG		rp_fact_balance_var
	inputWeightDTGMax		rp_goal_balance_min
	hvRatio		rp_goal_balance_avg
			rp_goal_balance_var
			h_ff_ratio

Table 4: List of features from the feature selection. The complete set of features is listed in Appendix A.

the features. Only 3 algorithms have statistically better accuracy with f-34 dataset and nine of them have the similar accuracy, but in all cases they were below the best achieved accuracy

3.2.4 REGRESSION MODELS

We have trained regression models only with the positive instances of the classification training phase. In the classification phase, all the planners have the same proportion of instances, but in this case, not all the planners have the same number of instances given that they solved a different number of problems. Nevertheless we do not consider this a relevant bias because the models include the planner name, which somehow encodes single models for each planner, but in a grouped model. We have trained the models with 20 regression algorithms, also provided by Weka.

The best algorithm for f-all was Decision Table (Kohavi, 1995) with a Relative Absolute Error (RAE) of 49.87 and the best one for f-34 was Bagging (Breiman, 1996) with a RAE of 50.62. Nevertheless, for simplicity we have selected the Decision Table model for the regression task in both datasets (f-all and f-34). This decision is justified because the results do not show a significant difference with the t-test result. In following sections, the regression model will always refer to that trained with the Decision Table algorithm. See all the results of the regression models in the Table 15 of Appendix B.

3.3 IBaCoP Strategies

We have considered various strategies for the configuration of the IBACOP portfolios. The list of the strategies is ordered depending on the use they make of the knowledge provided by the predictive models. In the experiments, each configuration will run for 1800 seconds. We have named the portfolios according to the names given in IPC-2014.

IBACOP**:** This portfolio uses an equal time strategy (ET) on the set of 11 candidate planners previously filtered by the QT-Pareto Score Filtering procedure. Therefore, the single planners will run for 163 seconds. This strategy does not use the predictive models. The planner using this strategy was awarded runner-up in the sequential satisficing track of IPC-2014.

IBACOP2**:** This portfolio uses the Best N confidence strategy (BN), where $N = 5$. This means that the 5 planners with the best prediction confidence in solving the problem are included in the configuration. The run time is assigned uniformly to each planner (360 seconds). This strategy, using the f-34 model was the winner of the sequential satisficing track of IPC-2014.[1]

IBACOP2-B5E**:** This portfolio uses the Best estimated time strategy (BNE) with $N = 5$. It follows the same procedure as IBACOP2 to select 5 planners, and then the time is assigned by scaling the time prediction provided by the regression model (Decision Table). This strategy participated in the learning track of IPC-2014 under the name of LIBACOP2. In this case the training data and models were generated for each domain separately, since the learning track provides a training problem set for each domain "a priori".

In addition, we have built other portfolio configurations that will serve as the baseline for comparison.

Overall Equal Time (OET): This strategy is a non-informed strategy which does not carry out any planner filtering or use predictive models. It assigns equal time for each available planner. Given that we have 28 planners (all the participants of IPC-2011 plus LPG-td), each planner will run for 64 seconds. With this planner we see the need for some planner filtering since, although it already obtains results close to current state of the art base planners, these results can be improved by selecting a reduced set of planners.

Best 11 Planners (B11): This strategy selects the top 11 planners of IPC-2011 ordered by the score in the competition, as shown in Table 2. Although selecting the best 11 planners is a good choice intuitively, we show in the table that this selection reduces the planner diversity in the portfolio, since most top planners in the competition are based on FD, with the only exception of Probe. This strategy is comparable with that implemented in BUS portfolio (Howe, Dahlman, Hansen, Scheetz, & von Mayrhauser, 1999), in which the control strategy for ordering the planners and allocating time is derived from the performance study data.

Random 5 Planners (Rand): This strategy is one of the baselines to compare to the best 5 confidence strategy (IBACOP2). Given a planning task, this strategy takes a random sample of 5 planners from the population of 11 candidate planners selected by the QT-Pareto filtering,

1. Predictive models submitted with IBACOP2 to IPC-2014 were trained in a different benchmark set. In that case the best accuracy was achieved by a Random Forest (Breiman, 2001).

and assigns equal time to them. We expect that a wise selection of 5 planners (IBACOP2) will be on average better than a random selection.

Default 5 Planners (Def): In this case, the strategy always includes the 5 best planners in terms of quality score over the training data. These 5 planners are a subset from the 11 candidate planners selected by the QT-Pareto filtering (i.e., LAMA-2011, PROBE, FD-AUTOTUNE-1, LAMA-2008 and FD-AUTOTUNE-2). Then, the time is assigned equitably. We want to see whether using the best 5 planners is better than making a per-instance selection of 5 planners.

3.4 Other Implementation Details

In this section we describe some of the engineering details we have incorporated into IBACOP planners. For instance, the competition rules proposed to include domains with conditional effects. Because of this, we have included a parser that translates tasks with conditional effects into an equivalent planning task without this property. This translator was based on a previous translator ADL2STRIPS (Hoffmann, Edelkamp, Thiébaux, Englert, dos Santos Liporace, & Trüg, 2006). Specifically, we have implemented the compilation that creates artificial actions for effect evaluations (Nebel, 2000).

Furthermore, many of the 11 candidate planners were built on the FAST-DOWNWARD framework, which among other things, separate the planning process into the sub-process of translation, pre-processing and search. Indeed, the translation and the pre-process steps are already executed when the feature generation for a given task is performed. We take advantage of this fact to avoid doing the first two steps repeatedly if some of these planners are included in the configuration of the portfolio for the regarding task. For version compatibility reasons this procedure is divided into two groups. The output of the FD pre-process, used for feature extraction, is also used as the search input for LAMA-2011, FDSS-2 and FD-AUTOTUNE (1 & 2). The previous FD pre-processor [2] was used in common for LAMA-2008, ARVAND and LAMAR. This optimization is used by all the strategies evaluated. The remaining planners are totally independent of the FD pre-processing.

Moreover, some bugs arose during the execution of IPC-2014, as some issues in the domain models required updates (Vallati, Chrpa, & McMcluskey, 2014a), and some planners were updated such as Mercury (Vallati, Chrpa, & McMcluskey, 2014b). These issues were also fixed prior to running the experimental evaluation presented in this article.

4. Experimental Evaluation

In this section, we describe the settings of the experimental evaluation and present the results of the planners on the benchmarks used in the IPC-2014, specifically, in the Sequential Satisficing track. In addition, we provide an analysis of the diversity of the planner selection achieved by some configurations.

4.1 Experimental Settings

We have evaluated the different portfolio strategies described in Section 3.3, which permits different portfolio configurations to be created. IBACOP2 and IBACOP2-B5E were run with two predictive model versions, one trained with all features (f-all) and the other one trained with the selected fea-

2. This version corresponds to the version used to submit planners to IPC-2011

tures (f-34). The Random strategy was run for 5 times and the average is reported. In addition, we have included the JASPER and MERCURY planners in the comparison. These planners also competed in IPC-2014. MERCURY (Domshlak, Hoffmann, & Katz, 2015) was the second best planner in terms of IPC score and JASPER (Xie, Müller, & Holte, 2014) was the second best planner in terms of problems solved (coverage). As the test set we have used all the benchmarks of IPC-2014, with the updates described in Section 3.4. This test set comprises 14 domains with 20 problems for each domain.

Experiments were run on a cluster with Intel XEON 2.93 Ghz nodes, each with 8 GB of RAM, using Linux Ubuntu 12.04 LTS. All planners had a cutoff of $1,800$ seconds and 4 GB of RAM. For IBACOP configurations requiring feature extraction, this process was limited to 4 GB of RAM (following IPC competition rules) and 300 seconds (which is the maximum time used in the training set to obtain the features, as described in Table 3). The time to extract the features is included in the execution of the portfolio where, in the worse case, the feature extraction process took 300 seconds and, therefore, the candidate planners only have $1,500$ to run. If the system does not extract the features in this time, the input features are treated as missing values.

4.2 Results

Table 5 shows the results of all evaluated planners using the IPC quality score. We recall that this score gives the ratio $\frac{Q^*}{Q_i}$ to planner i for each problem, where Q_i is the quality of the best solution found by planner i, and Q^* is the best solution found by any planner. If planner i does not solve the problem the score is 0.

								IBaCoP2		IBaCoP2-B5S	
	Mercury	Jasper	OET	B11	Def	Rand	IBaCoP	f-all	f-34	f-all	f-34
Hiking	18,9	18.1	18.2	**19.2**	18.7	18.4	19.0	18.9	18.6	18.8	18.6
Openstacks	**19.6**	18.8	15.4	17.2	19.2	16.3	17.8	18.6	18.5	18.2	18.3
Thoughtful	12.7	16.4	14.5	**19.4**	19.2	17.4	19.2	19.2	17.4	17.6	19.2
GED	**19.4**	17.9	18.3	17.1	16.3	13.0	17.5	17.6	17.5	17.6	17.5
Barman	14.6	**19.0**	16.7	16.7	17.2	13.8	16.9	17.1	17.1	17.2	17.2
Parking	18.0	17.0	17.6	13.8	18.0	11.6	16.3	18.1	18.1	**18.5**	18.1
Visitall	**20.0**	15.4	13.3	8.1	13.7	15.0	15.2	18.0	18.0	18.0	18.0
Maintenance	5.1	10.0	15.0	**15.9**	11.6	14.5	15.6	15.5	15.4	15.5	15.4
Tetris	**16.3**	16.2	5.0	11.5	9.3	11.9	13.3	12.5	11.9	15.7	13.6
Childsnack	0.0	0.0	12.0	3.4	2.6	8.9	**19.2**	18.4	18.9	15.0	18.9
Transport	**19.9**	12.0	8.9	3.8	6.9	8.2	10.3	11.5	11.6	11.1	12.1
Floortile	2.0	2.0	4.8	3.4	4.1	12.3	16.2	15.3	17.2	**17.5**	12.0
CityCar	4.1	11.5	6.0	8.8	5.0	9.4	**12.5**	9.0	6.2	9.9	7.78
CaveDiving	7.0	**8.0**	0.0	0.0	7.0	7.0	6.3	7.0	7.0	7.0	7.0
Total	177.6	182.1	165.7	158.3	168.8	177.6	215.3	216.5	213.4	**217.4**	213.7

Table 5: Results of IBACOP configurations. The table also includes the results of Jasper, Mercury and the four baseline configurations, OET, Best 11, Default and Random.

The overall best planner was IBACOP2-B5E (f-all), closely followed by IBACOP2 (f-all). The difference between these two configurations is negligible. All the configurations using predictive models are much better than OET, Default, Best 11 or Random. IBACOP has a very good performance, comparable to the best performance. Moreover, there is a big difference between our configurations and the other planners (Jasper and Mercury). IBACOP based configurations are 32 or more points higher in all cases.

Figure 2 details the evolution of the number of problems solved as a function of the run-time elapsed. The far right-hand point of the figure represents the final coverage. The best planner in terms of coverage is IBACOP, with 249 problems, and the second is IBACOP2 (f-all) with 246. In Figure 2, the planners show two different behaviors. On the one hand, an asymptotic growing in the number of problems solved demonstrates that giving more time to the planners does not permit the number of problems solved to be increased. JASPER is an extreme case, which after 300 seconds is almost unable to improve. MERCURY has the same problem, as well as the portfolio configurations that do not take care of diversity. However, the IBACOP, IBACOP2 and IBACOP2-B5E, which selected a diverse set of planners, show a growing behavior throughout the time.

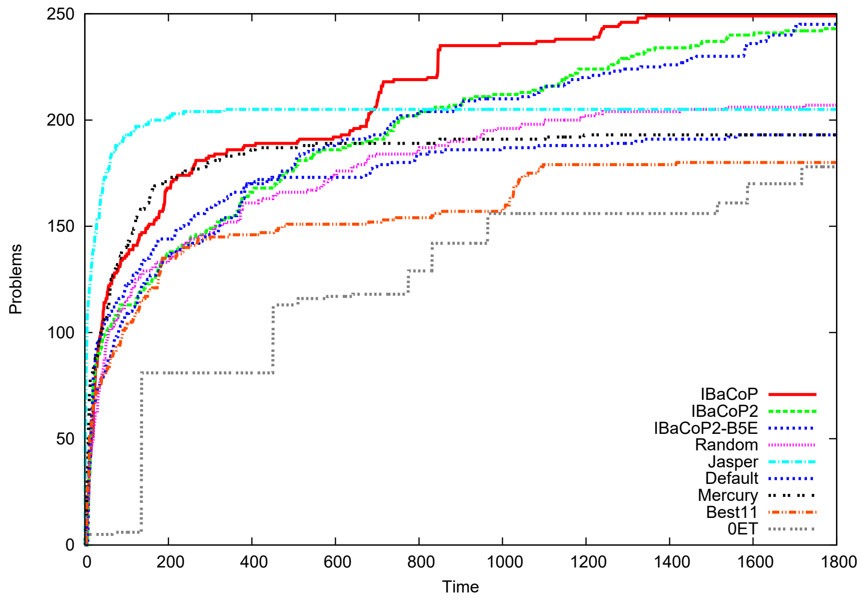

Figure 2: Comparison of IBACOP configurations, the baseline configurations, and the Mercury and Jasper planners.

From the results we can derive some insights regarding different configurations. The score difference between OET and IBACOP reveals the importance of making a pre-selection of candidate planner with an accurate filtering procedure. The Pareto-dominance approach allows us to have a smaller set of planners, which means having more time per planner. There is a trade-off between having more time per planner and loosing the diversity of solvers, and the results suggest that it is more important to maintain diversity than increasing running time per planner. For instance, the 11 best IPC-2011 planners (B11) obtain worse results than those using the original 28 (OET), even though B11 base planners have a longer running time. However, the QT-Pareto filtering approach is able to reduce the number of planners while not sacrificing the diversity, which produces very good results.

Reducing the number of planners for the portfolio configuration from 11 to 5 puts in risk the diversity of solvers, as shown in the results of the Def approach (the best 5 planners in terms of performance) or in Rand (the random selection of 5 planners). Nevertheless, IBACOP2 (f-all) and (f-34) perform quite better than Def and Rand, which demonstrates that the classification models select on average a good subset of planners for solving each particular task. These results are quite promising for exploiting empirical performance models in planning portfolios. However, in the current setting, results of IBACOP2 are quite similar to IBACOP. Thus, the classification models manage to reduce the set of planners without deteriorating the performance of the fixed portfolio, but they hardly contribute to a better overall performance.

Table 6 presents the number of problems solved by each of the 11 candidate planners. The final column has the maximum number of problems that can be solved by the complete set of candidate planners (i.e., a problem can be solved only if at least one of the candidate planners solved the problem). The optimal selection of 5 planners for each planning task would lead to 253 problems solved. IBACOP2 is close to this optimum, confirming its ability for selecting good candidates for the portfolio. The default configuration solved 193 problems, and the average number of problems solved by the random configuration is 207 problems. Both of them are far from the best possible value.

	lama11	probe	FDA1	lama08	FDA2	lamar	arvand	fdss2	ya2-mt	LPG	M	Max
Hiking	18	**20**	18	**20**	**20**	**20**	**20**	**20**	4	**20**	3	**20**
Thoughtful	15	12	16	17	12	14	**20**	17	13	8	5	**20**
Openstacks	**20**	4	19	**20**	**20**	**20**	**20**	12	0	1	0	20
Tetris	9	14	15	8	1	13	**18**	17	0	0	0	**18**
GED	**20**	**20**	**20**	0	0	0	0	**20**	0	14	0	**20**
Transport	15	12	7	12	6	7	5	10	**20**	0	0	**20**
Parking	**20**	9	14	13	2	18	0	16	0	0	0	**20**
Barman	**20**	19	15	13	2	15	0	8	0	0	0	**20**
Maintenance	7	8	10	1	8	1	**17**	16	3	8	6	**17**
CityCar	1	0	5	4	5	8	**19**	5	2	0	14	**19**
Visitall	**20**	10	0	2	0	0	2	0	**20**	1	0	**20**
Childsnack	0	0	2	2	0	2	8	3	0	7	**20**	**20**
Floortile	2	2	2	2	5	2	1	2	0	**19**	0	**19**
CaveDiving	0	0	0	0	0	0	0	0	0	0	**7**	**7**
total	166	130	143	114	81	120	128	146	62	74	55	**260**

Table 6: Results of the candidate planners defined in Table 1 and the maximum number of problems that can be solved by the complete set of these planners.

Once the set of 5 planners has been selected for the per-instance configuration, the regression models do not contribute to a better performance. The task of estimating the run time needed to solve a problem is more difficult than the classification task (Schwefel, Wegener, & Weinert, 2013). Additionally, given that the aggregated time predictions could exceed the time limit, our proposal rescales these estimations and alters the real predictions. One alternative to this proposal is to keep the real prediction and run the planners in the order established by the confidence in the classification prediction, until one of them reaches the time limit. However, preliminary experiments during the development of the planner showed us that this approach does not compensate the risk of losing diversity due to fewer planner executions.

Another aspect to be analyzed is the performance of the planners in the new domains. The IPC-2014 incorporated seven new domains, which means that the QT-Pareto Filtering and the predictive models have not been trained with them. These domains are Cave Diving, Child-Snack, CityCar, GED, Hiking, Maintenance and Tetris. From the results we can conclude that the behavior of all IBACOP configurations in new domains is on average similar to the performance in previously seen domains.

4.3 Per-Instance Selection of Planners

In the previous section we showed that the benefit of configuring a portfolio per problem is that the set of selected planners can be better adjusted to the problem, using fewer planners, and providing more execution time to each planner. In this section we want to analyze the diversity of the planner selections made by IBACOP2 to see if the predictive models are classifying planners by how good they are at solving specific domains or if they are identifying properties of specific problems in different domains. Note that the test problems of a given domain usually range from easy to hard. The increase in difficulty is mainly due to a larger size of the problems. Nevertheless, this increase affects the learning features at a different scale and intensity.

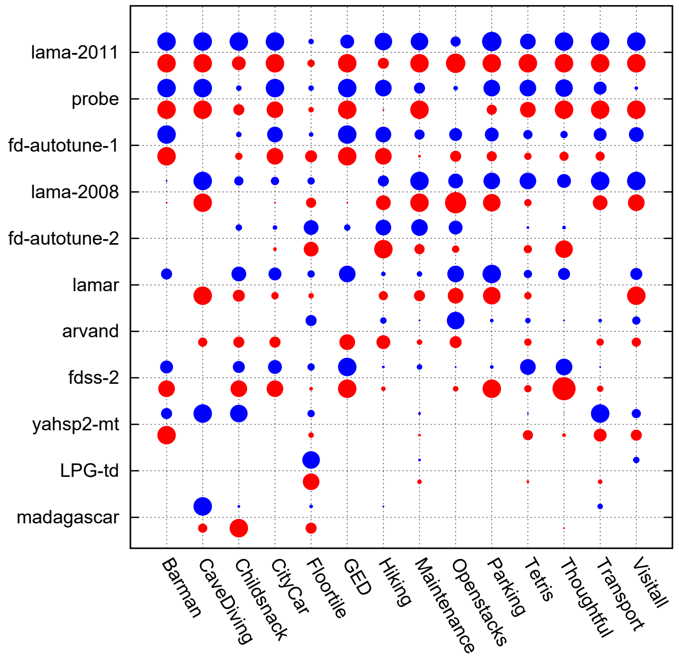

Figure 3: Proportion of the number of times each planner has been selected in a domain. In red dots, the proportion for IBACOP2 (f-all), and in blue dots, the proportion for IBACOP2 (f-34).

Figure 3 shows the diversity of planners according to the selection made by IBACOP2 (blue dots for f-34 and red dots for f-all). The x axis shows the IPC-2014 domains and the y axis lists the 11 candidate planners that the portfolio can use. The size of the dots is proportional to the number of times a planner has been selected for a particular domain, i.e. the number of problems for which the planner was selected. If a domain has five dots in one column (one domain), it means that it was selected by the portfolio configuration for all problems in the domain. However, every column with more than five dots reveals the use of different 5-planner sets for different problems in the same domain. The highlight of this analysis is that the 11 planners have been selected in at least one domain, and in 13 out of 14 domains the selections involve more than 5 planners. Note, for instance, that LAMA-2011 has the best "a priori" confidence on solving problems, but it is sometimes not used (i.e., it was selected only 6 times in Floortile and 11 times in Openstacks). Furthermore, some planners have a low "a priori" probability of being selected, but are frequently used in some domains (like LPG-TD in Floortile).

Table 7 shows the sum of the number of times that each planner has been selected. The maximum number of times that a planner could be selected is $14 \times 20 = 280$. The last column reports the average and the standard deviation of the number of times that each planner has been selected per domain in both approximations (all and the reduced set of features).

	f-34	f-all	Average	±	STD
LAMA-2011	248	256	18,00	±	4,02
PROBE	200	206	14,50	±	6,71
FD-AUTOTUNE-1	173	151	11,57	±	6,29
LAMA-2008	173	157	14,50	±	6,71
FD-AUTOTUNE-2	93	88	6,46	±	7,13
LAMAR	152	133	10,18	±	6,98
ARVAND	65	111	6,29	±	5,90
FDSS-2	122	149	9,68	±	7,94
YAHSP2-MT	95	71	5,93	±	7,26
LPG-TD	29	31	2,14	±	4,99
MADAGASCAR	35	45	2,86	±	5,73

Table 7: Number of times a candidate planner has been selected by the two different classification models (f-34 and f-all).

In addition to the previous analysis, we wanted to delve into the underlying mechanism to achieve the per-instance selection of planners. We recall that planners are selected based on the confidence of the success prediction. Therefore, in order to achieve different 5-planner sets in the same domain, the ranking of the prediction confidence should vary throughout the problem. To visualize and confirm this fact, we have selected the Tetris domain, which is one of the new domains in IPC-2014 and it shows a good diversity selection as shown in Figure 3. This domain is a simplified version of the well-known Tetris game.

A heatmap with the success prediction confidences appears in Figure 4. At a glance we realized that in general, a planner with higher success rate in training time obtains higher confidence, but confidence ranking varies throughout different problems of the same domain. Another way to read the picture is that the 5 darkest squares per column form the set of selected planners. For instance, lama-2011 was selected in all problems and probe was selected 18 times. On the other hand Madagascar was not selected, and LPG-td was selected 3 times.

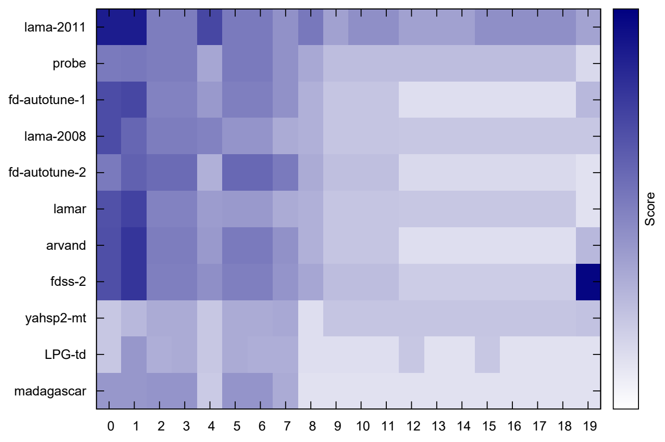

Figure 4: Success prediction confidence provided by the classification model (f-all) for each planner and problem in the tetris Domain. Scale goes from 0.0 (white) or no confidence at all to 1.0 (dark blue) or complete confidence.

5. Related Work

In this section, we summarize the relevant research into portfolio configuration and how it relates to our work. In addition, we summarize different approaches for the characterization of the planning tasks, which is a cornerstone of this work to predict the behavior of the planners.

The idea of exploiting the synergy of different solvers to improve the performance of the individual ones is applied in propositional satisfiability problems (SAT), constraint satisfaction problems (CSP), answer set programming (ASP) and in the scope of this paper, Automated Planning. The SAT area has carried out extensive research into the importance of selecting the components of the portfolio (Xu, Hutter, Hoos, & Leyton-Brown, 2012) and how to select each component (Lindauer, Hoos, Hutter, & Schaub, 2015b) automatically. The study of strategy selection in this area includes per-instance selections (Lindauer, Hoos, & Hutter, 2015a). In addition, there is an intensive study into solver runtime prediction (Hutter, Xu, Hoos, & Leyton-Brown, 2015), including a good characterization of the satisfiability task. In other fields of Artificial Intelligence, CSP has portfolio configurations based on machine learning techniques such as SUNNY (Amadini, Gabbrielli, & Mauro, 2014b) and other empirical research (Amadini, Gabbrielli, & Mauro, 2014a). For example in ASP, the ASP-based Solver Scheduling (Hoos, Kaminski, Schaub, & Schneider, 2012) is a multi-criteria optimization problem and provides the corresponding ASP encodings. In this paper we only report the main systems related to Automated Planning in detail.

5.1 Portfolios in Automated Planning

Howe et al. (1999) describes one of the first planner portfolios. They implement a system called *BUS* that runs 6 planners and whose goal is to find a solution in the shortest period of time. To achieve it, they run the planners in portions of time and in circular order until one of them finds a solution. In this portfolio, the planners are sorted following the estimation provided by a linear regression model of their success and run-time so, as in our case, they use predictive models of the behavior of the planners to decide their order of execution. However, they use only 5 features extracted from the PDDL description. For the domain, they count the number of actions and the number of predicates. For the problem, they count the number of objects, the number of predicates in the initial conditions and the number of goals. *BUS* minimizes the expected cost of implementing a sequence of algorithms until one works, in contrast to IBACOP and IBACOP2, that does not stop until the assigned time is over.

Fast Downward Stone Soup (FDSS, Helmert et al., 2011) is based on the Fast Downward (FD) planning system (Helmert, 2006), with several versions for the different tracks. *FDSS* is an approach to select and combine heuristics and search algorithms. A configuration is a combination of a search algorithm and a group of heuristics. In training, they evaluate the possible configurations with a time limit, and select the set of configurations that maximizes the coverage. For the portfolio presented in the IPC-2011 Sequential Satisficing Track, they sort the configurations by decreasing the order of coverage, hence beginning with algorithms likely to succeed quickly. The time limit for each component is the lowest value that would still lead to the same portfolio score in the training phase. However, the order is important, since each setting communicates the quality of the best solution found so far to the following one, and this value is used to improve the performance of the next setting. Therefore, *FDSS* can only include configurations within the FD framework. Conversely, IBACOP and IBACOP2 build a portfolio using a mixture of generic planners of different styles and techniques. Indeed *FDSS* is one of IBACOP candidate planners.

PbP (Gerevini et al., 2009) configures a domain-specific portfolio. This portfolio incorporates macro-actions in the specific knowledge of the domains. The incorporation of this knowledge establishes the order of a subset of planners which contain macro-actions. The running time is assigned through a round-robin strategy. This portfolio incorporates seven planners (the latest version, *PbP2*, adds lama-2008, see Gerevini et al., 2014). The automatic portfolio configuration in *PbP* and IBACOP aims to build different types of planning systems: a domain-optimized portfolio planner for each given domain in *PbP* and IBACOP is an efficient domain-independent planner portfolio. The IBACOP and *PbP* configuration processes are significantly different. *PbP* uses several planners that focus on macro-actions whilst IBACOP only uses generic planners. The execution scheduling strategy of *PbP* runs the selected planners in round-robin rather than sequentially in the case of IBACOP.

Fast Downward Cedalion (Seipp, Sievers, Helmert, & Hutter, 2015) is an algorithm for automatically configuring sequential planning portfolios of a parametric planner. Given a parametric planner and a set of training instances, it selects the pair of planner and time iteratively. At the end of each iteration all instances for which the current portfolio finds the best solution are removed from the training set. The algorithm stops when the total run time of the added configurations reaches the portfolio time limit or if the training set becomes empty. Configurations are generated using the SMAC (Hutter, Hoos, & Leyton-Brown, 2011) model-based algorithm configurator on the remaining training instances. *Cedalion* has the same configuration for all the problems but a different configuration per version and IBACOP has a different configuration per problem. The diversity of the candidate planner is limited while IBACOP may completely include independent base planners. The configuration processes and the resulting configured portfolios of *Cedalion* are the same as *FDSS*.

The Fast Downward Uniform (Seipp et al., 2012) portfolio runs 21 automatically configured Fast Downward instantiations sequentially for the same amount of time. Uniform portfolio approaches are configured using the automatic parameter tuning framework ParamILS (Hutter, Hoos, Leyton-Brown, & Stützle, 2009) to find fast configurations of the Fast Downward planning system for 21 planning domains separately. At runtime, all configurations found are run sequentially for the same amount of time for at most 85 seconds.

MiPlan (Núñez et al., 2015) is a sequential portfolio using Mixed-Integer Programming, which computes the portfolio that obtains the best achievable performance with respect to a selection of training planning tasks. In their case they have created a sequential portfolio with a subset of sequential planners with fixed times whilst IBACOP2 has different configurations per problem. For this approximation, the planner does not consider the other portfolios, only their components. In contrast, IBACOP and IBACOP2 includes the planners as they appear in other competitions, i.e. as black boxes.

5.2 Features in Planning Problems

The construction of models to predict the performance of planners is not a novel idea. Roberts et al. (2008, 2009) showed that models learned from the planners performance on known benchmarks up to 2008 obtain a high accuracy when predicting whether a planner will succeed or not. They use 19-32 features extracted from the domain and problem definition. The main difference with our approach is that we also include features based on SAS$^+$, the heuristics of the initial state and the fact balance of the relaxed plan. Most of our features come from the ground instantiation of the problem, which are the key to differentiate tasks that share the same feature values at the PDDL level.

Torchlight (Hoffmann, 2011) is a toolkit which allows the search space topology to be analyzed without actually running any search. The analysis is based on the relation between the topology under delete relaxation heuristics and the causal graph as well as DTGs. The feature extraction process is built on top of the FF planner (Hoffmann & Nebel, 2001).

Recently, Fawcett et al. (2014) has generated models for accurately predicting the planner run time. These models exploit a large set of instance features, including many of the features depicted in Section 2.1.2. These features are derived from the PDDL and SAS$^+$ representations of the problem, a SAT encoding of the planning problem and short runs of planners. Some other features are extracted with Torchlight (Hoffmann, 2011). The experimental results in the work indicate that the performance models generated are able to produce very accurate run time predictions. This study of empirical performance models has not been applied to portfolio configurations.

6. Conclusion and Future Work

In this work we have introduced a framework for the creation of configurable planning portfolios, IBACOP. In the first step of the portfolio creation we find a small number of planners that maintains the diversity of the initial planner set based on the QT-Pareto score filtering. Then we train predictive models that select a promising sub-set of planners for solving a particular planning task.

The experimental evaluation confirmed the great performance of IBACOP and IBACOP2 in IPC-2014. We can summarize the lessons learned from the development of the current IBACOP portfolios as the following:

- What really matters in the generation of a good portfolio is the selection of a diverse set of planners. We have shown that the QT-Pareto score filtering reduces the set of candidate planners while preserving the diversity. This filtering produces better results than other rankings based on coverage or quality score.

- The selection of smaller sets of planners for the portfolio configuration (e.g., a sub-set of 5 planners in our experiments) is dangerous given that the portfolio might lose planner diversity. We observed this situation in the Def and Random configurations, which select 5 out of 11 planners.
- The portfolio configurations using the classification models are able to select a good subset of 5 planners, which with uniformly distributed time outperformed the selection provided by a random and default selection with the same number of planners.
- Estimating the runtime for solving a problem is still very difficult and for this reason regression models are not providing additional useful information for the portfolio construction.
- In their current form, predictive models hardly contribute to the overall performance of the portfolio. Per-instance configurations using classification models achieve similar performance to the fixed portfolio, but running fewer planners.

Even though in the current architecture the benefits of using predictive models are limited, the results are promising because of the good performance of IBACOP2 compared to the baseline configurations. We think there is some room for research in this direction. Our argument is that static portfolio configurations (including IBACOP) are limited by the components and the fixed time bound for each base planner. Their performance has an upper-limit, as computed by MiPlan, that is smaller than the achievable performance of a dynamic configuration. This is because in a per-instance configuration the portfolio strategy could assign different times to the base planners.

As future work we want to study additional features for a better characterization of the planning tasks. Any computation that could be carried out as a pre-process step, or even with information on first evaluated search nodes, could help with making predictive models more accurate. Our models could incorporate information, for instance, about the landmark graph or the time elapsed in computing the initial state heuristics. Other future work is a study of importance of the created features, including a comparison between different groups of them in accordance with the semantics of the features.

7. Acknowledgments

We thank the authors of the base planners because our work is based largely on their previous effort. This work has been partially supported by the Spanish projects TIN2011-27652-C03-02, TIN2012-38079-C03-02 and TIN2014-55637-C2-1-R.

A. Appendix: Complete Feature Description

In this Appendix we present the list of the features used to characterize a planning task. For each feature we include a brief description of what it is or how it is computed. Features are grouped by their category in separate tables.

A.1 PDDL Features

N.	Name	Description
1	*Objects*	The number of objects in the problem.
2	*Goals*	The number of goals in the problem.
3	*Init*	The number of in facts in the initial state.
4	*Types*	The number of types in the domain.
5	*Actions*	The number of actions in the domain.
6	*Predicates*	The number of predicates in the domain.
7	*Axioms*	The number of axioms in the domain.
8	*Functions*	The number of functions in the domain.

Table 8: PDDL Features.

A.2 FD Instantiation Features

N.	Name	Description
9	*Relevant facts*	The number of facts marked as relevant by FF instantiation.
10	*Cost metric*	Whether action costs are used or not.
11	*Generated rules*	The number of created rules in the translation process to create SAS$^+$ task.
12	*Relevant atoms*	The number of relevant atoms found in the translator process.
13	*Auxiliary atoms*	The number of auxiliary atoms found in the translator process.
14	*Final queue length*	The length of the queue at end of the translation. This queue is an auxiliary list that is used in the translation process to compute the model.
15	*Total queue pushes*	The number of times an element has been pushed into the queue.
16	*Implied effects removed*	The number of implied effects removed. Where the implied effects that the translator knows are already included.
17	*Effect preconditions added*	The number of implied effects added.
18	*Translator variables*	The number of created variables in SAS$^+$ formulation.
19	*Derived variables*	The number of state variables that correspond to derived predicates or to other artificial variables not directly affected by operator applications.
20	*Translator facts*	The number of facts that the pre-process takes into account.
21	*Mutex groups*	The number of mutex groups.
22	*Total mutex size*	The sum of all mutex group sizes.
23	*Translator operators*	The number of instantiated operators in SAS$^+$ formulation.
24	*Total task size*	The allowed memory for the translation process.

Table 9: Features extracted from the console output of the FD system.

A.3 SAS$^+$ Feature Description

We recall that in CG, the *high-level* variables are the variables for which there is a defined value in the goal. Although the common definition of the CG does not consider the edges as weighted, the FD system computes the edge weights of the CG as the number of instantiated actions that induced each edge. We also consider these weights for computing our features.

N.	Name	Description
General Features		
25	*Number Variables*	The number of variables of the CG.
26	*High-Level Variables*	The number of high-level variables.
27	*TotalEdges*	The number of edges.
28	*TotalWeight*	The sum of the edge weights.
CG Ratios		
29	*VERatio*	The ratio between the total number of variables and the total number of edges. This ratio shows the level of connection in the CG.
30	*WERatio*	The ratio between the sum of the weights and the number of edges. This ratio shows the average weight for the edges.
31	*WVRatio*	The ratio between the sum of the weights and the number of variables.
32	*HVRatio*	The ratio between the number of high-level variables and the total number of variables. This ratio shows the percentage of variables involved in the problem goals.
Statistics of the CG		
33-35	*InputEdge*	Maximum, average and standard deviation of the number of incoming edges for each variable.
36-38	*InputWeight*	Maximum, average and standard deviation of the sum of the weights of the incoming edges for each variable.
39-41	*OutputEdge*	Maximum, average and standard deviation of the number of outgoing edges for each variable.
42-44	*OutputWeight*	Maximum, average and standard deviation of the sum of the weights of the incoming edges for each variable.
Statistics of high-level Variables		
45-47	*InputEdgeHV*	The number of incoming edges for each of the high level variables. This value produces three new features following the same computation as *InputEdgeCG* (features 33-35).
48-50	*InputWeightHV*	The edge weight sum of the incoming edges for each of the high level variables. This value produces three new features following the same computation as *InputWeightCG*.
51-53	*OutputEdgeHV*	The number of outgoing edges for each of the high level variables.
54-57	*OutputWeightHV*	The sum of the weights of the incoming edges for each high level variables.

Table 10: Features from the Causal Graph.

N.	Name	Description
	General Aggregated Features DTG	
58	*Number Vertices*	The sum of the number of nodes of all DTGs.
59	*Total Edges*	The sum of the number of edges of all DTGs.
60	*Total Weight*	The sum of the edge weights of all DTGs. The edge weight in a DTG corresponds to the cost of applying the action that induced the edge.
	DTG Ratios	
61	*edVa Ratio*	The ratio between the total number of edges and the total numbers of variables. This ratio shows the level of connection in the DTG.
62	*weEdRatio*	The ratio between the sum of the weights and the number of edges. This ratio shows the number of restrictions that need to make the transition.
63	*weVaRatio*	The ratio between the sum of the weights and the number of variables.
	Statistics of DTGs	
64-66	*Input Edge*	Maximum, average and standard deviation of the number of incoming edges for a vertex in a DTG.
67-69	*Input Weight*	Maximum, average and standard deviation of the sum of the weights of the incoming edges of all nodes.
70-72	*Output Edge*	Maximum, average and standard deviation of the number of outgoing edges for a vertex in a DTG.
73-75	*Output Weight*	Maximum, average and standard deviation of the sum of the weights of the outgoing edges of all nodes.

Table 11: Features that aggregate the information from the DTGs.

A.4 Heuristic Features

N.	Name	Description
76	*Max*	(Bonet, Loerincs, & Geffner, 1997; Bonet & Geffner, 2000) The maximum of the accumulated costs of the paths to the goal propositions in the relaxed problem.
77	*Landmark cut*	(Helmert & Domshlak, 2009) The sum of the costs of each disjunctive action landmark that represents a cut in a justification graph towards the goal propositions.
78	*Landmark count*	(Richter, Helmert, & Westphal, 2008) The sum of the costs of the minimum cost achiever of each unsatisfied or required again landmark.
	Goal count	The number of unsatisfied goals.
79	*FF*	(Hoffmann & Nebel, 2001) The cost of a plan that reaches the goals in the relaxed problem that ignores negative interactions.
80	*Additive*	(Bonet et al., 1997; Bonet & Geffner, 2000) The sum of the accumulated costs of the paths to the goal propositions in the relaxed problem.
81	*Causal Graph*	(Helmert, 2004) The cost of reaching the goal from a given search state by solving a number of sub problems of the planning task which are derived from the causal graph.
82	*Context-enhanced additive*	(Helmert & Geffner, 2008) The causal graph heuristic modified to use pivots that define contexts relevant to the heuristic computation.

Table 12: Unit cost heuristics included as features.

A.5 Fact Balance

N.	Name	Description
83-85	*RP_init*	Minimum, average and variance of the number of times that a fact in the initial state is deleted in the computation of the relaxed plan.
86-88	*RP_goal*	Minimum, average and variance of the number of times that a goal is deleted in the computation of the relaxed plan.
89	*Ratio_ff*	The ratio between the value of the max and FF heuristic. This proportion shows the idea of parallelization of the plan.

Table 13: Fact balance features.

B. Appendix: Learning Results

This appendix shows the detailed results for the machine learning algorithms used to train the predictive models.

B.1 Classification

Algorithm	f-all dataset			f-34 dataset			
rules.ZeroR	61.72	±	0.03	61.72	±	0.03	
rules.Ridor	82.52	±	2.48	81.76	±	2.11	
rules.PART	90.81	±	0.89	89.62	±	0.89	•
rules.JRip	87.21	±	1.38	86.26	±	1.20	
rules.DecisionTable	85.78	±	0.98	84.94	±	1.37	
rules.ConjunctiveRule	69.33	±	1.20	69.64	±	1.61	
trees.REPTree	89.08	±	0.85	88.06	±	0.89	•
trees.RandomTree	86.39	±	1.81	87.91	±	0.95	◦
trees.RandomForest	90.96	±	0.78	90.27	±	0.85	•
trees.LMT	91.11	±	0.72	90.03	±	0.94	•
trees.J48	90.84	±	1.01	89.24	±	0.87	•
trees.ADTree	75.46	±	1.24	74.39	±	1.30	•
trees.NBTree	90.38	±	0.88	89.47	±	0.92	•
trees.DecisionStump	67.96	±	0.96	64.10	±	1.30	•
lazy.LWL	67.96	±	0.96	63.48	±	1.86	•
lazy.IBk -K 1	85.93	±	0.84	82.97	±	1.03	•
lazy.IBk -K 3	86.04	±	0.90	84.13	±	1.03	•
lazy.IBk -K 5	85.36	±	0.91	84.17	±	1.01	•
meta.RotationForest	**93.39**	±	**0.70**	**92.35**	±	**0.73**	•
meta.AttributeSelectedClassifier	89.69	±	0.89	88.64	±	1.00	•
meta.ClassificationViaClustering	52.32	±	1.98	57.99	±	2.66	◦
meta.ClassificationViaRegression	90.82	±	0.84	89.80	±	0.75	•
meta.Bagging	90.99	±	0.74	89.83	±	0.85	•
meta.MultiClassClassifier	77.15	±	1.09	75.02	±	1.14	•
functions.SimpleLogistic	76.37	±	1.12	74.48	±	1.23	•
functions.MultilayerPerceptron	87.27	±	1.65	88.65	±	1.01	◦
functions.RBFNetwork	67.71	±	1.03	68.10	±	1.17	
functions.SMO	75.39	±	1.16	73.94	±	1.10	•
bayes.NaiveBayes	69.00	±	0.98	68.87	±	0.97	
bayes.NaiveBayesUpdateable	69.00	±	0.98	68.87	±	0.97	
bayes.BayesNet	75.43	±	1.29	75.05	±	1.21	

Table 14: Accuracy and standard deviation for each training algorithm using 10-fold cross-validation. Also, results of a t-test (O'Mahony, 1986) for the two training sets is shown. Symbols ◦, • means statistically significant improvement or degradation respectively. The significance level in the t-test is 0.05 and the baseline is the left column.

B.2 Regression

	f-all dataset		f-34 dataset		
	RAE	ρ	RAE	ρ	
trees.DecisionStump	82.09 ± 2.36	0.42 ± 0.05	82.09 ± 2.36	0.42 ± 0.05	
trees.REPTree	57.70 ± 3.40	0.66 ± 0.05	56.69 ± 3.36	0.67 ± 0.05	
trees.RandomTree	59.28± 6.06	0.55± 0.07	53.71± 4.54	0.61± 0.06	○
trees.RandomForest	52.54± 2.66	0.71 ±0.04	45.62± 2.68	0.76 ± 0.03	○
functions.M5P	60.44 ± 13.26	0.59 ± 0.18	56.38 ± 4.22	0.65 ± 0.09	
rules.ConjunctiveRule	87.31 ± 2.79	0.38 ± 0.06	87.25 ± 2.80	0.39 ± 0.06	
rules.DecisionTable	**49.87 ± 3.03**	0.69 ± 0.04	**51.19 ± 2.78**	0.68 ± 0.05	
rules.M5Rules	90.60 ± 138.25	0.58 ± 0.18	65.84 ± 12.74	0.61 ± 0.14	
meta.Bagging	**50.95 ± 2.71**	0.74 ± 0.04	**50.62 ± 2.58**	0.74 ± 0.04	
meta.AdditiveRegression	80.91 ± 3.21	0.51 ± 0.04	79.93 ± 3.29	0.51 ± 0.04	
lazy.IBk 1	92.96 ± 11.09	0.36 ± 0.06	66.73 ± 5.17	0.54 ± 0.06	○
lazy.IBk 3	74.31 ± 6.31	0.47 ± 0.06	63.57 ± 4.25	0.60 ± 0.05	○
lazy.IBk 5	73.03 ± 5.91	0.47 ± 0.06	64.38 ± 3.81	0.60 ± 0.05	○
lazy.KStar	69.26± 3.35	0.44± 0.05	67.75± 3.36	0.47±0.05	
lazy.LWL	81.82± 2.30	0.43± 0.05	81.82± 2.33	0.43± 0.05	
functions.LinearRegression	77.71 ± 2.55	0.55 ±0.04	78.58± 2.52	0.51± 0.04	
functions.MultilayerPerceptron	86.01± 72.86	0.66 ±0.05	81.59± 45.93	0.66 ±0.05	
functions.LeastMedSq	66.36 ± 2.94	0.33 ± 0.08	66.29 ± 3.01	0.31 ± 0.07	
functions.RBFNetwork	94.20±1.60	0.23± 0.05	94.25±1.54	0.21±0.04	
functions.SMOreg	57.01 ± 2.88	0.48 ± 0.05	58.75 ± 2.62	0.45 ± 0.05	

Table 15: Results for the 10-fold cross-validation in the regression models. RAE is the Relative Absolute Error and ρ is the correlation coefficient. The small RAE values are better. Symbols ○, ● means statistically significant improvement or degradation respectively. The significance level in the t-test is 0.05 and the baseline is the left column.

C. Appendix: Training Results

t

	L-11	Probe	FDA1	L-08	FDA2	Lamar	Arvand	FDSS2	ya2-mt	LPG	M	#
openstacks	30	30	30	30	30	30	27	30	0	27	11	30
pathways	30	30	26	29	29	30	30	0	0	30	30	30
rovers	40	40	40	40	40	40	40	40	40	40	40	40
storage	40	40	40	40	40	40	40	40	40	40	40	40
tpp	30	30	30	30	30	30	30	30	30	24	30	30
trucks	19	8	18	16	22	15	15	20	0	11	21	30
pipesworld	42	44	40	38	33	43	46	42	41	33	14	50
cybersec	28	24	28	28	26	27	28	28	0	7	0	30
Openstacks-adl	31	31	31	31	31	31	31	31	0	1	16	31
openstacks	30	30	30	30	30	30	30	30	1	0	15	30
pegsol	30	30	30	30	30	30	30	30	22	1	27	30
scanalyzer	30	30	30	30	27	30	30	30	27	0	21	30
sokoban	29	27	29	25	27	25	8	29	0	0	2	30
transport	18	10	17	17	18	17	19	15	11	0	9	30
woodworking	23	30	25	26	24	25	30	30	23	0	2	30
elevators	30	29	30	25	30	27	30	30	2	0	0	30
barman	20	20	20	4	6	6	0	17	12	0	0	20
elevators	20	20	20	6	17	11	20	20	0	0	0	20
floortile	6	5	7	3	9	3	3	7	8	12	0	20
nomystery	10	6	10	12	19	12	19	12	10	0	17	20
openstacks	20	14	20	20	20	20	20	19	0	2	0	20
parcprinter	20	14	20	1	14	0	20	20	13	0	20	20
parking	20	19	19	20	9	20	4	20	3	0	0	20
pegsol	20	20	20	20	20	20	20	20	15	0	17	20
scanalyzer	20	20	20	20	17	20	20	20	17	0	11	20
sokoban	19	17	19	15	16	14	2	19	0	0	0	20
tidybot	16	18	15	14	17	19	17	18	0	15	1	20
transport	19	20	11	19	10	3	15	15	20	0	0	20
visitall	20	20	2	20	5	11	10	6	20	8	0	20
woodworking	20	20	20	14	14	9	20	20	19	0	1	20
Gold-miner	30	30	30	30	26	29	30	0	30	30	30	30
Matching-bw	25	15	24	23	23	17	16	0	25	22	1	30
N-puzzle	29	20	30	29	9	27	6	0	20	30	0	30
parking	28	24	25	28	16	30	17	0	13	13	0	30
sokoban	23	23	30	18	30	17	30	30	28	15	22	30
thoughtful	0	18	0	0	0	0	0	0	22	7	0	30
barman	9	5	0	0	0	0	0	13	0	0	0	30
blocksworld	29	30	22	21	15	0	0	20	16	29	0	30
depots	1	30	0	0	0	6	0	0	29	6	0	30
gripper	0	0	0	0	30	0	4	0	0	30	0	30
parking	18	9	6	13	1	19	4	9	0	0	0	30
rovers	30	30	30	29	24	30	30	30	30	11	14	30
satellite	16	10	3	3	29	1	2	22	13	30	0	30
spanner	0	0	0	0	0	0	0	0	0	30	15	30
tpp	30	20	30	30	6	21	30	25	30	1	9	30
Total	998	960	927	877	869	835	823	837	630	505	436	1251

Table 16: Solved problems in the training phase. The first part in this table is the results of IPC-2005, the second part IPC-2008 and IPC-2011 in the satisficing tracks. The two last rows (from Gold-miner to tpp) are IPC-2008-2011 in the learning track. The last column is the number of problems included for training.

D. Appendix: Planners

The following list the set of planners pre-selected as candidates from the Pareto-dominance filtering described in Section 3.1

- Arvand (Nakhost, Müller, Valenzano, & Xie, 2011): is a stochastic planner that uses Monte Carlo random walks to balance exploration and exploitation in heuristic search. This version uses an online learning algorithm to find the best configuration of the parameters for the given problem.
- Fast Downward Autotune-1 and Fast Downward Autotune-2 (Fawcett, Helmert, Hoos, Karpas, Röger, & Seipp, 2011): are two instantiations of the FD planning system automatically configured for performance on a wide range of planning domains, using the well-known ParamILS configurator. The planners use three main types of search in combination with several heuristics.
- Fast Downward Stone Soup-2 (Helmert et al., 2011) (FDSS-2): is a sequential portfolio with several search algorithms and heuristics. Given the results of the training benchmarks, the best combination of algorithms and heuristics is found through a hill-climbing search. Here, the only information communicated between the component solvers is the quality of the best solution found so far.
- LAMA-2008 and LAMA-2011 (Richter & Westphal, 2010; Richter, Westphal, & Helmert, 2011) is a propositional planner based on the combination of landmark count heuristic and FF heuristic. The search performs a set of weighted A^* with iteratively decreasing weights. The planner was developed within the FD Planning System (Helmert, 2006).
- Lamar (Olsen & Bryce, 2011) is a modification of the LAMA planner that includes a randomized construction of the landmark count heuristic.
- Madagascar (Rintanen, 2011): implements several innovations of SAT planning, including compact parallelized/interleaved search strategies and SAT-based heuristics.
- Probe (Lipovetzky & Geffner, 2011): exploits the idea of wisely constructed lookaheads or *probes*, which are action sequences computed without searching from a given state that can quickly go deep into the state space, terminating either in the goal or in failure. This technique is integrated within a standard greedy best first search.
- YAHSP2-MT (Vidal, 2011) extracts information from the relaxed plan in order to generate lookahead states. This strategy is implemented in a complete best-first search algorithm, modified to take helpful actions into account.
- LPG-td (Gerevini et al., 2006) is based on stochastic local search in the space of particular action graphs derived from the planning problem specification.

References

Amadini, R., Gabbrielli, M., & Mauro, J. (2014a). Portfolio approaches for constraint optimization problems. *TPLP*, *8426*, 21–35.

Amadini, R., Gabbrielli, M., & Mauro, J. (2014b). SUNNY: a lazy portfolio approach for constraint solving. *TPLP*, *14*(4-5), 509–524.

Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial intelligence*, *97*(1), 245–271.

Bonet, B., & Geffner, H. (2000). Planning as heuristic search: New results. In *Recent Advances in AI Planning*, pp. 360–372. Springer.

Bonet, B., Loerincs, G., & Geffner, H. (1997). A robust and fast action selection mechanism for planning. In *AAAI/IAAI*, pp. 714–719.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Cenamor, I., de la Rosa, T., & Fernández, F. (2012). Mining IPC-2011 results. In *Proceedings of the Third Workshop on the International Planning Competition - ICAPS*.

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

Censor, Y. (1977). Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization*, *4*(1), 41–59.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In Kittler, J., & Roli, F. (Eds.), *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000, Proceedings*, Vol. 1857 of *Lecture Notes in Computer Science*, pp. 1–15. Springer.

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

Fawcett, C., Helmert, M., Hoos, H., Karpas, E., Röger, G., & Seipp, J. (2011). FD-Autotune: Domain-specific configuration using fast-downward. *Proceedings of the Workshop on the Planning and Learning - ICAPS*, *2011*(8).

Fawcett, C., Vallati, M., Hutter, F., Hoffmann, J., Hoos, H. H., & 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)*.

Gerevini, A., Saetti, A., & Vallati, M. (2009). An automatically configurable portfolio-based planner with macro-actions: PbP. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09)*.

Gerevini, A., Saetti, A., & Serina, I. (2006). An approach to temporal planning and scheduling in domains with predictable exogenous events. *Journal of Artificial Intelligence Research*, *25*, 187–231.

Gerevini, A., Saetti, A., & Vallati, M. (2014). Planning through automatic portfolio configuration: The PbP approach. *Journal of Artificial Intelligence Research*, *50*, 639–696.

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated planning: theory & practice*. Access Online via Elsevier.

Gomes, C. P., & Selman, B. (2001). Algorithm portfolios. *Artificial Intelligence*, *126*(1), 43–62.

Grabczewski, K., & Jankowski, N. (2005). Feature selection with decision tree criterion. In *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems (HIS05)*, pp. 212–217. IEEE.

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: concepts and techniques*. Elsevier.

Helmert, M. (2004). A planning heuristic based on causal graph analysis. In *In Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, Vol. 16, pp. 161–170.

Helmert, M. (2006). The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, *26*, 191–246.

Helmert, M. (2009). Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, *173*, 503–535.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway?. In *In Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09)*.

Helmert, M., & Geffner, H. (2008). Unifying the causal graph and additive heuristics. In *In Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS-08)*, pp. 140–147.

Helmert, M., Röger, G., Seipp, J., Karpas, E., Hoffmann, J., Keyder, E., Nissim, R., Richter, S., & Westphal, M. (2011). Fast downward stone soup. *The Seventh International Planning Competition*, *IPC-7 planner abstracts*, 38.

Hoffmann, J. (2003). The metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, *20*, 291–341.

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.

Hoffmann, J., Edelkamp, S., Thiébaux, S., Englert, R., dos Santos Liporace, F., & Trüg, S. (2006). Engineering benchmarks for planning: the domains used in the deterministic part of IPC-4. *Journal of Artificial Intelligence Research*, *26*, 453–541.

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

Hoos, H., Kaminski, R., Schaub, T., & Schneider, M. T. (2012). aspeed: ASP-based solver scheduling. *ICLP (Technical Communications)*, *17*, 176–187.

Howe, A. E., Dahlman, E., Hansen, C., Scheetz, M., & von Mayrhauser, A. (1999). Exploiting competitive planner performance. In Biundo, S., & Fox, M. (Eds.), *Recent Advances in AI Planning, 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999, Proceedings*, Vol. 1809 of *Lecture Notes in Computer Science*, pp. 62–72. Springer.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pp. 507–523. Springer.

Hutter, F., Hoos, H. H., Leyton-Brown, K., & Stützle, T. (2009). ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, *36*, 267–306.

Hutter, F., Xu, L., Hoos, H., & Leyton-Brown, K. (2015). Algorithm runtime prediction: Methods and evaluation (extended abstract). In Yang, Q., & Wooldridge, M. (Eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 4197–4201. AAAI Press.

Kohavi, R. (1995). The power of decision tables. In *Machine Learning: ECML-95*, pp. 174–189. Springer.

Linares López, C., Celorrio, S. J., & Olaya, A. G. (2015). The deterministic part of the seventh international planning competition. *Artificial Intelligence*, *223*, 82–119.

Lindauer, M. T., Hoos, H. H., & Hutter, F. (2015a). From sequential algorithm selection to parallel portfolio selection. In Dhaenens, C., Jourdan, L., & Marmion, M. (Eds.), *Learning and Intelligent Optimization - 9th International Conference, LION 9, Lille, France, January 12-15, 2015. Revised Selected Papers*, Vol. 8994 of *Lecture Notes in Computer Science*, pp. 1–16. Springer.

Lindauer, M. T., Hoos, H. H., Hutter, F., & Schaub, T. (2015b). Autofolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, *53*, 745–778.

Lipovetzky, N., & 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)*, pp. 154–161.

Malitsky, Y., Sabharwal, A., Samulowitz, H., & Sellmann, M. (2013). Algorithm portfolios based on cost-sensitive hierarchical clustering. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pp. 608–614. AAAI Press.

Nakhost, H., Müller, M., Valenzano, R., & Xie, F. (2011). Arvand: the art of random walks. *The Seventh International Planning Competition*, *IPC-7 planner abstracts*, 15–16.

Nebel, B. (2000). On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research*, *12*, 271–315.

Núñez, S., Borrajo, D., & Linares López, C. (2015). Automatic construction of optimal static sequential portfolios for AI planning and beyond. *Artificial Intelligence*, *226*, 75–101.

Olsen, A., & Bryce, D. (2011). Randward and Lamar: Randomizing the FF heuristic. *The Seventh International Planning Competition*, *IPC-7 planner abstracts*, 55.

O'Mahony, M. (1986). *Sensory evaluation of food: statistical methods and procedures*, Vol. 16. CRC Press.

Quinlan, J. R. (1993). *C4. 5: programs for machine learning*, Vol. 1. Morgan kaufmann.

Richter, S., Helmert, M., & Westphal, M. (2008). Landmarks revisited. In *AAAI*, Vol. 8, pp. 975–982.

Richter, S., & Westphal, M. (2010). The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, *39*(1), 127–177.

Richter, S., Westphal, M., & Helmert, M. (2011). Lama 2008 and 2011. *The Seventh International Planning Competition*, *IPC-7 planner abstracts*, 50.

Rintanen, J. (2011). Madagascar: Efficient planning with SAT. *The Seventh International Planning Competition*, *IPC-7 planner abstracts*, 61.

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

Roberts, M., Howe, A. E., Wilson, B., & desJardins, M. (2008). What makes planners predictable?. In *In Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS-08)*, pp. 288–295.

Rodriguez, J. J., Kuncheva, L. I., & Alonso, C. J. (2006). Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(10), 1619–1630.

Schwefel, H.-P., Wegener, I., & Weinert, K. (2013). *Advances in computational intelligence: Theory and practice*. Springer Science & Business Media.

Seipp, J., Braun, M., Garimort, J., & Helmert, M. (2012). Learning portfolios of automatically tuned planners. In McCluskey, L., Williams, B., Silva, J. R., & Bonet, B. (Eds.), *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*. AAAI.

Seipp, J., Sievers, S., Helmert, M., & Hutter, F. (2015). Automatic configuration of sequential planning portfolios. In Bonet, B., & Koenig, S. (Eds.), *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pp. 3364–3370. AAAI Press.

Vallati, M. (2012). A guide to portfolio-based planning. In *Multi-disciplinary Trends in Artificial Intelligence*, pp. 57–68. Springer.

Vallati, M., Chrpa, L., & Kitchin, D. E. (2015). Portfolio-based planning: State of the art, common practice and open challenges. *AI Communications*, *29*, 1–17.

Vallati, M., Chrpa, L., & McMcluskey, L. (2014a). *Competition Domains*. https://helios.hud.ac.uk/scommv/IPC-14/benchmark.html.

Vallati, M., Chrpa, L., & McMcluskey, L. (2014b). *Source code and Erratum Deterministic part*. https://helios.hud.ac.uk/scommv/IPC-14/errPlan.html.

Vidal, V. (2011). YAHSP2: Keep it simple, stupid. *The Seventh International Planning Competition, IPC-7 planner abstracts*, 83.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd Edition, Morgan Kaufmann.

Xie, F., Müller, M., & Holte, R. (2014). Jasper: the art of exploration in greedy best first search. In *Planner abstracts*. IPC-2014.

Xu, L., Hoos, H., & Leyton-Brown, K. (2010). Hydra: Automatically configuring algorithms for portfolio-based selection. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, pp. 210–216.

Xu, L., Hutter, F., Hoos, H., & Leyton-Brown, K. (2012). Evaluating component solver contributions to portfolio-based algorithm selectors. In *Theory and Applications of Satisfiability Testing–SAT 2012*, pp. 228–241. Springer.

Xu, L., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2008). SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, *32*, 565–606.