Learning Predictive Models to Configure Planning Portfolios

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

Abstract

One of the latest advances for solving classical planning problems is the development of new approaches such as portfolios of planners. In a portfolio, different base planners are run sequentially to solve a problem. Therefore, the main challenge of a portfolio planner is to define what base planners to run, in what order, and for how long. This configuration can be created manually or automatically, for instance, using machine learning techniques. In this work, a dynamic portfolio planner is described which, opposite to previous portfolio planners, is able to adapt itself to every new problem. The portfolio automatically selects the planners and the time according to predictive models. These models estimate whether a base planner will be able to solve the problem and, if so, how long it will take. The predictive models are created with machine learning techniques, using the data of the last International Planning Competition (IPC). Prediction capability of the models depend on the features extracted from the IPC results for each problem. In this work, we use a group of features extracted from the SAS+ formulation of such problems. We define different portfolio strategies, and we show that the resulting portfolios provide an improvement when compared not only with the winning planner of the last competition (LAMA), but also with less informed portfolio strategies.

Introduction

The International Planning Competition (IPC) is an excellent initiative to foster the study and development of automated planning systems. Since the event takes the shape of a competition with different tracks, after the event, a planner is selected as winner of each track. Different planning systems have dominated the competition in different years. However, one of the main invariants of the competition is that there is not a single planner which is always better (nor at least equal) than the other planners for every problem. This means that, although there is a planner which, following the quality metrics of the competition, can be considered the best one, we can always find some problems in different domains where other planners outperform the global winner.

The idea of using a set of base systems to generate solutions more accurate than the ones obtained separately is not new in Artificial Intelligence. For instance, in machine learning, meta-classifiers use different base classifier systems 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 can outperform the results of single search strategies (Xu et al. 2007).

In automated planning, the portfolios of planners have taken the interest of the community. A portfolio planner can be defined as a set of planners with a selection strategy. Such selection strategy has to define three main elements: (1) **what sub-set of planners to run**, (2) **how long to run each planner?** and (3) **in what order**. In this work we propose to answer the previous questions using Machine Learning. Specifically, we use the results of the sequential satisfying track of the IPC 2011 to construct predictive models about the capability of the base planners to solve planning problems -first question-, as well as the time that they require -second question. The order in which the planners are executed is given by the confidence of the predictive models obtained. With those predictive models, we are able to define a different portfolio configuration for each planning problem, similarly to previous works about the use of portfolios in search, but a novelty in automated planning where previous works always have focused in static portfolios (Gerevini, Saetti, and Vallati 2009; Gagliolo and Schmidhuber 2006). From the machine learning point of view, defining an accurate set of features to characterize the planning problem is critical. Specifically, we use data extracted from three different sources. Firstly, the IPC-2011 software (López 2011) contains several packages, which facilitates testing planners, compare their performance and obtain reports of the results. IPCReport is the package in charge of providing access to the data generated during the competition, so we used this software to extract the results of every planner in every problem of the competition. Secondly, some basic features can be obtained from the PDDL problem files, like the number of literals, objects or goals of a problem, which gives an idea of the size of the problems. Lastly, additional features are extracted from the graphs induced by the SAS+ formalism (Backstrom and Nebel 1995; Helmert 2009) in order to partially recognize the differences between problems of similar size (Cenamor, de la Rosa, and Fernández 2012).

We propose to evaluate the resulting portfolio estimating the behavior that we can expect in the future. For this performance estimation we have to consider whether the problems

belong or not to one of the domains used during the learning of the predictive models. Therefore we suggest to use two evaluation strategies derived from the machine learning literature, split and leave-one-domain-out, as will be explained later.

The remainder of the paper is organized as follows. In the next section we present the predictive models of planner performance, where we will explain the learning process followed to obtain such models, describing the features that we use in this work. Following that, we describe how we create the portfolio, and the different strategies to configure the portfolio. Afterwards, we describe the empirical evaluation of the portfolios. The paper finishes with the related work, the conclusions and the future research lines.

Learning Predictive Models of Planner Performance

Constructing the planning portfolio and learning the predictive models is described from a Data Mining perspective, as shown in Figure 1. Data Mining is a process of discovering implicit knowledge from determinate data. This process may contain different phases depending on the goals. In our case, we have defined the data mining goals as the creation of two predictive models. First, whether a planner will be able to solve a problem, and if so, what will be the time required to compute the best plan. The first problem is a classification task, where the predicted attribute is a Boolean: the planner solved the problem or not. The second problem is a regression task, where the output belongs to the positive real numbers, but restricted to the time limit given to the planners (i.e., 1800 seconds in IPC). The reason why we have chosen these two tasks is two-fold. On the one hand, we want to characterize under which conditions a planner will succeed, so this characterization will support a better knowledge of the planners and their possible improvement (Cenamor, de la Rosa, and Fernández 2012). On the other hand, and from a more engineering point of view, we want to obtain predictive models that can be used for the selection of planners when configuring the portfolio-based planner.

The first part of the work flow of the mining process is the gathering of the features from the planning problems. Given the problems of the last IPC (IPC-2012), a subset of features is extracted using the software developed by the organizers. The report of this software presents a lot of variables to principal observations about the execution of the planners for a given problem and domain. Among all these variables we used the name of the planner, the domain, the problem, a list of a time solutions and a list of a quality solutions. These two last variables represent all the solutions for a problem in a given planner sorted by appearance order. From all the data of the IPC 2011, we used the problems of the sequential satisfying track. We got 7560 instances, corresponding to the execution of 27 planners in a total of 20 problems for 14 domains. There are 3837 positive instances, i.e., executions returned a plan, and 3723 negative instances, i.e., no plan was returned from such execution.

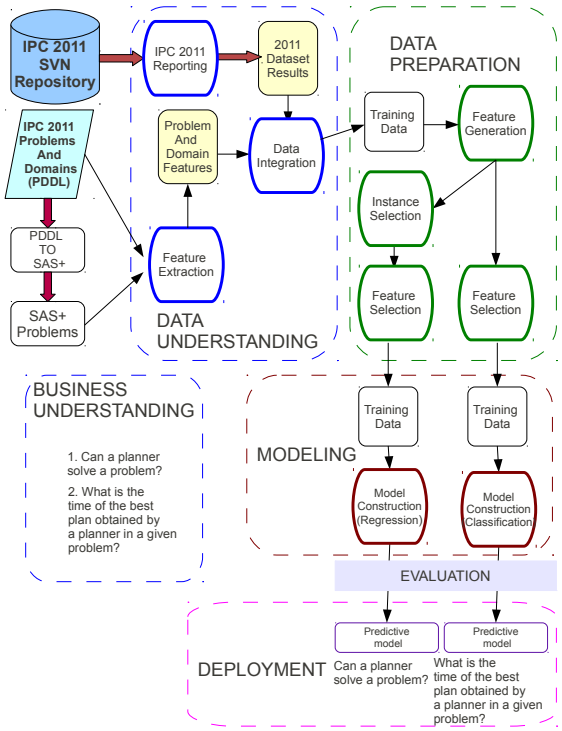

Figure 1: Data work-flow of the mining process following CRISP-DM (Chapman et al. 2000) methodology

Besides, we used the problems of this track to create some features that can characterize the problems (without consider their solutions). In order to obtain a good characterization we used features extracted from the PDDL files and a set of elaborated features generated from the problem translation to the SAS+ formalism and its induced graphs, i.e., causal graphs and domain transition graphs. The basic features (from PDDL) are typically features of the planning problem: number of *objects* defined in the problem, number of instantiated predicates in the initial state (*literals*) and number of instantiated predicates that are true in the final state (*goals*). The SAS+ formalism is an alternative representation to STRIPS (Backstrom and Nebel 1995; Helmert 2009). Using this formalism, a problem instance can be represented in a structured way using two types of graphs: The first is the ***causal graph*** (CG), which is a graph that captures the causal dependencies between the state variables of a given problem. The second is the ***domain transition graph*** (DTG) which encodes the allowed transitions between different values of a variable. In a problem there is a DTG for each state variable. For more details see (Helmert 2006).

We have used the LAMA planner (Richter and Westphal 2010) to pre-process and generate all the graphs. We recall that in the causal graph, the *high-level* variables are the variables for which there is a defined value in the goal. Although the common definition of the causal graph does not consider

the edges as weighted, LAMA computes the edge weights of the causal graph as the number of instantiated actions that induced each edge. We also consider these weights for computing our features. We have extracted a total of 47 features for each problem, which are summarized next.

Feature Description

For the CG we generated features in four categories: (1) general, which includes the direct information from the graph; (2) ratios, which represents interesting proportions that may be equal across problems of different size; (3) statistical, such as the average, maximum and the standard deviation of the entire graph; and (4) high-level statistical, the same as before but only considering the high-level variables.

The general variables of a CG are four: the number of variables, the number of high-level variables, the number of edges and the sum of weights of the edges. The ratios are four: The first is the ratio between the total number of variables and the total edges. The second is the ratio between the sum of the weights and the number of variables. The third is the ratio between the number of high-level variables and the total number of variables. And the last is the ratio between the number of high-level variables and the total number of variables.

The statistical information of a CG is used to characterize the structure of the causal graph. We compute the average, the maximum and the standard deviation of the following four values: The first is the number of incoming edges for each variable. The second is the sum of the weights of the incoming edges for each variable. The third is the number of outgoing edges for each variable. And the last is the sum of the weights of the incoming edges for each variable.

The statistical information of high-level variables is used to encode the structure for the variables involved in the problem goals. We compute the same as the statistical information of the CG of the following two values: the number of incoming edges for each variable, and the sum of the weights of the incoming edges for each variable.

For the DTG we generated features in two categories: (1) general, aggregating the relevant properties of all graphs and (2) general aggregated features and some statistics over all graphs.

The general variables of the DTG are three: the number of variables, the number of edges and the sum of weights of the edges. The statistical information of the DTG is used to characterize the structure of the whole domain transition graph. In this case, we compute the same statistical information as for the CG, but in this case when we compute the average, it is the average of all the graphs. In the case of standard deviation, we compute the standard deviation of all the graphs and the same with the maximum.

Once we have read the SAS+ problem, the computation time to extract all those features is inconsiderable because we only realize sums, averages and standard deviation computations.

Data Preparation

After the extraction of the features, data preparation is typically the following mining step. In this phase, we create the output features for learning the models. The first task is to learn whether a planner will be able to solve a problem. This problem is a classification task with a binary class. This attribute is set to "yes" if there exists at least one solution of the problem; otherwise it is set to "no". In this case, the quality of the solution is not relevant.

The second task is to learn the time that a given planner expended in a given problem. This attribute is a numerical attribute in the range $[0..1800]$, limits defined by the IPC competition. In this case, we have eliminated all the instances where the a planner was not able to find a solution, i. e. where time and quality vector are missing.

Data Modeling

The data modeling is divided into two parts as defined above: generating a classification model and generating a regression model. The classification model is a decision tree created by J48 algorithm (Quinlan 1993), although we performed tests with other algorithms like decision rules (PART) (Frank and Witten 1998), Support Vector Machines (SMO) (Cristianini and Shawe-Taylor 2000), and IBK (Witten and Frank 2005) for different values of k $(1, 3, 5)$. The implementation of these algorithms is provided by WEKA (Witten and Frank 2005), and they are used with the pre-defined parameters.

The regression model is created by instance-based learning (Briscoe and Caelli 1996) (IBK) with $k = 3$. However, like in the classification case, we used other algorithms like the decision trees for regression problems (M5Rules) (Wang and Witten 1996), IBK (Witten and Frank 2005) for different values of k $(1, 5)$ and Support Vector Machines in regression (SMOreg) (Shevade et al. 2000). The implementation of these algorithms is also provided by WEKA.

Evaluation

We follow two different evaluation mechanisms to estimate the behavior of the models in different circumstances. The first way to evaluate the performance of a model is to split the available data in two sets: a training set and a test set. In our case, we divide the problems in two sets depending on its identifier: even or odd [1]. We constructed two models: one with even problems, which is then evaluated with the odd ones and vice versa. The resulting is the sum of both processes. In this way, we can evaluate how the model constructed will behave in previously unseen problems.

The second evaluation mechanism permits to evaluate how the models will behave in problems of unseen domains. The approach is based in the leave-one-out evaluation method, which in machine learning can be seen as a cross-validation [2] where k is set to the number of available

[1] The problems of the competition are created in increasing difficulty, so separating them in this way almost ensures that the difficulty of the problems in the two sets generated is very similar, as the results will show.

[2] Cross-validation (Browne 2000) permits to estimate the classification accuracy (percentage of times that the model outputs the expected class) of a classifier in the future, or the predicting capability (relative absolute error of the predicted value respect to the

data. In our case, the method is a cross-validation where the data is not separated in folds randomly, but per domains. Therefore, with this approach we create as many folds as domains and, each time, we build a predictive model with the data from all the domains except one. In this way we estimate the behavior of the learned models in previously unseen planning domains.

Data Exploitation

Table 1 shows how to use the predictive models learned (as shown in Section) to build a portfolio of planners.

Predictive Models Based Portfolio
• Given
1. An input vector, d, which represents all the relevant features of a planning domain
2. An input vector, pr, which represents all the relevant features of a planning problem
3. A set of planners $P = \{pl_1, \ldots, pl_n\}$
4. A maximum execution time, t
5. A predictive Model $C(pl, d, pr) \rightarrow < s, c >$ that for any planner, pl, domain d, and problem, pr outputs whether pl will solve problem pr, s, and what is the confidence, c, of such estimation
6. A predictive Model $R(pl, d, pr) \rightarrow < t, e >$ that for any planner, pl, domain d, and problem, pr, outputs the estimated time that pl will require to find the best solution of pr, and what is the standard error expected in that estimation
1. $eligible = \emptyset$
2. for $i = 1$ to n do
(a) $< s_i, c_i >= C(pl_i, d, pr)$
(b) if $s_i == true$ then $eligible = eligible \cup pl_i$
3. For $j = 1$ to $\|eligible\|$, $< t_j, e_j >= R(pl_j, d, pr)$
4. Use the set $eligible$ and the predictive estimations, $< s_j, c_j >$ and $< t_j, e_j >$, to create the portfolio, following any selection strategy
5. Execute the portfolio

Table 1: Algorithm to create a portfolio based on the predictive models.

The method assumes that it receives all the relevant features of the planning problem in a given domain encapsulated in vectors d and pr. It also receives the set of planners, the maximum execution time, and the predictive models.

Then, for each of the n planners, the algorithm obtains from the classification model the estimation of their capability to solve the problem. All the planners whose answer is positive are included in the set of $eligible$ planners. For all the planners in $eligible$ the estimated time to be run is also estimated with the regression model. The output of the algorithm is the configuration of the portfolio, i.e. a list of planners with an associated run time. The portfolio is created using different strategies, which are defined in the following section. In case of the sum of the times of this list are larger than the limit t, the list is truncated.

expected one) of a regression model. A cross-validation splits the data randomly in k groups, $(k-1)$ used for training the model and the rest to test the learned model. This process is repeated k rounds. The result of this process is the average of the results obtained by all the models computed in the k rounds.

Building strategies with predictive models

We have evaluated various strategies for the configuration of the portfolio. The list of the strategies is ordered depending on the use that they make of the knowledge provided by the prediction models. The first one does not use such knowledge at all, while the last one uses both classification and regression models.

Equal Time (ET): This strategy does not use the predictive models. It assigns equal time for each planner (uniform strategy). This means that, if we have 27 planners (all the participants of IPC 2011), all the planners will run for $1800/27 = 66.67$ seconds.

Best Confidence Estimation (BCE): This strategy uses the classification model. It selects the planner that ensure that the problem will be solved, but only the planner with the maximum confidence. If the classification model estimated that no planner is be able to solve the problem, it chooses the planner with a lower confidence of fail. In case of a confidence tie, it chooses all the planners in the tie. The execution time is also distributed uniformly among all the planners selected.

Best 5 Confidence (B5C): This strategy also uses the classification model. It selects the 5 planners with the highest confidence of solving the problem. The run time is assigned uniformly to each planner (360 seconds).

Best 10 Confidence (B10C): This strategy is equivalent to the previous one, but selecting 10 planners instead of 5, and therefore, assigning 180 seconds to each planner.

Best 5 Regression (B5R): This strategy uses the classification and regression models. It follows the same procedure than B5C to select 5 planners. Then, it estimates the total time required by the planners as the sum of the predicted run time of each planner. Since this sum is likely to be different from the maximum execution time (1800 seconds), the time assigned to each planner is a linear proportion with respect to the total time.

Best 10 Regression (B10R): This strategy is equivalent to B5R, but selecting 10 planners.

Experimental Results

In this section we explain the results of the models from their predictive capability point of view. The predictive power of the models is relevant because they give clues about wheter the portfolio strategies will success or fail. Then, we show and analyze the results of exploiting the models in the different portfolio strategies.

Estimated Performance of the Models Learned

Predictive models do not usually behave perfectly, i.e. a 100% of success in classification nor an error of zero in regression can be achieved. In fact, every data-set has a maximum performance, which is typically called the Bayesian optimal. The Bayesian optimal is produced by two reasons. The first one is noise and/or mistakes in the data; the second one is a lack of information which is required to improve the predictions. The performance of the models learned is shown in Tables 2 and 3 for different classification and regression algorithms tested, respectively. We show results following two different evaluation strategies, the classical split validation and leave-one-domain-out (both described above).

Data set	Split Validation	Leave One Domain Out
J48	**88.75 (1.05)**	59.14 (12.13)
IBk -K 1	88.67 (1.29)	60.83 (10.13)
IBk -K 3	87.63 (1.07)	60.58 (11.76)
IBk -K 5	88.58 (1.07)	**61.95 (11.10)**
SMO	72.48 (1.58)	61.34 (10.10)

Table 2: Classification accuracy and standard deviation for predicting planner success in the sequential satisfying track

The estimated performance of the classification models following the split validation is very high (close to a 90% of classification success). It is important to highlight that the data set is well-balanced in the class distribution: there are 3837 positive instances and 3723 negative ones, so a default classifier (i.e., a base classifier that always predicts the majority class) would obtain a performance of 50.75, while J48 obtains 88.75. Comparing the different algorithms tested, the best result is obtained using decision trees (J48), but IBk obtained similar accuracy. These results reveal that this model is good to predict the planner success for problems of already seen domains and will be very useful to build the portfolios under this scenarios (as will be described later).

We also have performed a brief automated feature selection process prior the generation of the models using the default parameters in WEKA. However, the process is very aggressive and eliminates most of the features, all except the planner and if the planner solved the problem or not (the class). The results with only those features are worse than with all the features, 72.06 ± 1.52 independently of the learning algorithm. We could perform a more extensive evaluation with additional feature selection processes and/or algorithms. However, we will show later that the models obtained at this point are good enough to build the portfolios.

In the case of the leave one domain out evaluation process, the results are worse, and a maximum performance of 61.95 is achieved with IBK. This result is 26.8 points worse than when evaluating with split validation, but still 10 points higher than the default classifier. The reason is that it is much more difficult to generalize to problems in new domains than to new problems in the same domains. In other words, the training data gathered from the 14 domains of the IPC 2011 is not a representative set of all the possible domains that can be modeled in PDDL. Anyway, we will show later that this result is promising.

Table 3 shows the results of different regression algorithms evaluated. The error metric used is the Relative Absolute Error (RAE), because it is independent of the range of values of the estimated function. The results obtained are around a 63%, which means that if, in average, the execution time were 100, in average we should make a mistake of 63 seconds. We will also show later that this value is good enough to provide successful estimates in the portfolios.

Algorithm	Split validation	Leave one Domain Out
M5Rules	73.66 (3.61)	985.64 (2200.93)
IBk -K 1	67.57 (4.07)	93.66 (23.38)
IBk -K 3	**62.98 (3.12)**	85.96 (22.26)
IBk -K 5	64.39 (3.00)	**85.57 (19.21)**
SMOreg	69.50 (2.87)	907.32 (2620.74)

Table 3: Relative absolute error and standard deviation of predicting the time that the planners will invest in finding the first, median and best solution in the sequential satisfying track

The best solution in all the cases is the algorithm IBk with $k = 3$ and $k = 5$ in split validation and leave one domain out. The model with lower error will be used in the portfolios. We follow a pessimistic approach, and the relative error is used in the regression strategies to assign the time. I.e. if the regression model estimates a run time of 100 seconds, we assign 163.

Performance of the Portfolios

In this section we evaluate two different generalization scenarios. The first one evaluates how a portfolio learned from some problems in different domains generalize to new problems in the same domains, or what we called above, the split evaluation. The second one evaluates how a portfolio learned from some problems in some domains generalize to problems in new domains. In both cases, problems used for training were not used in the test.

Table 4 shows the result of different portfolio strategies for the split evaluation. It also includes the results of LAMA-2011 and the best possible strategy (BS), both to have a reference for comparison. For each strategy we show the number of solved problems (S), the number of plans that have better quality than LAMA-2011 (+), and the number of problems that have worse quality than LAMA-2011 (-). The number of problems solved by BS is the number of problems solved in the track; therefore it is an upper bound for any conceivable portfolio configuration since we did not introduce new planners for our experiments. We can see that the best possible strategy would solve 267 problems and that 181 of them could have a better quality than the reported by LAMA-2011. That confirms there is a considerable room for improving the performance of the winner of the sequential satisfying track of IPC.

Table 4 shows the number of problems solved and the number of problems that obtain better and worse quality than LAMA-2011, respectively. The less informed strategy, *ET*,

	ET			BCE			B5C			B10C			B5R			B10R			Lama	BS	
	S	+	-	S	+	-	S	+	-	S	+	-	S	+	-	S	+	-	S	S	+
Barman	20	19	0	20	12	8	20	18	0	20	19	0	20	19	0	20	19	0	20	20	20
Elevators	20	16	2	20	14	6	20	17	2	20	20	0	20	16	2	20	19	0	20	20	20
Floortile	8	4	4	8	4	0	8	4	1	8	4	0	8	4	2	8	4	0	6	9	5
Nomystery	15	7	0	18	9	0	17	8	1	17	8	0	18	9	1	17	8	0	10	19	10
Openstacks	20	2	18	20	3	6	20	5	6	20	3	9	20	4	7	20	3	11	20	20	17
Parcprinter	20	0	20	20	8	2	20	8	1	20	11	0	20	8	1	20	11	0	20	20	11
Parking	12	3	16	20	0	20	20	1	16	20	4	12	20	1	16	20	4	13	20	20	9
Pegsol	20	0	8	20	0	2	20	0	2	20	0	2	20	0	2	20	0	2	20	20	0
Scanalyzer	18	2	14	19	9	5	18	8	4	17	8	6	18	8	6	18	10	3	20	20	13
Sokoban	17	5	10	18	2	6	19	4	1	18	4	2	19	4	1	19	5	1	19	19	6
Tibybot	16	5	9	18	6	5	19	6	7	18	7	4	17	4	7	17	6	4	16	20	13
Transport	20	9	11	20	11	9	19	10	8	20	14	6	19	10	8	20	14	6	19	20	18
Visitall	20	20	0	20	18	1	20	20	0	20	20	0	20	20	0	20	20	0	20	20	20
Woodworking	20	9	0	20	16	2	20	18	0	20	19	0	20	18	0	20	19	0	20	20	19
Total	246	101	112	**261**	112	72	260	127	49	258	141	41	259	125	53	259	**142**	**40**	250	267	181

Table 4: Comparison of the six portfolio strategies in split evaluation. Columns labeled with "S" show the number of problems solved in each domain. Columns labeled with "+" show the number of problems solved with plans of better quality to the ones reported by LAMA-2011. Columns labeled with "-" show the number of problems solved with worse quality than LAMA-2011.

executes every planner for a a fixed time. This strategies shows two important issues: on the one hand confirms that a portfolio is an interesting approach, since it is close to the winner of the competition. On the other hand, it shows that over $87,85\%$ of the problems are solved in less than 70 seconds (this strategy splits the time in 27 slices of 66.67 seconds).

The classification based strategies (BCE, B5E and B10C) shows that the classification models are useful, and they can solve over 96 % of the problems that can be solved (the limit is 267 problems). Classification based strategies solve more problems than the *ET* strategy. The use of the regression models to assign the execution time to each planner does not increment the number of problems solved over using only the classification models, and regression based strategies (B5R and B10R) solve a similar number of problems. The best portfolio of the competition, Fast Downward Stone Soup 2 (fdss-2), solved less problem than all the strategies: Fdss-2 solved 221 problems, and we solved in the worse case 258 problems.

Although quality is not estimated directly by the prediction models, the regression based strategies show that the regression models permit to maintain the quality of the problems obtained: given that the models predict the time to obtain the best solution, the regression models are accurate to compute an amount of time enough to achieve high quality solutions. All the informed strategies are better than *ET* because the difference in number of problems solved is at least 12 problems more, but they also improve the quality of the solutions. For instance, *B5C* improves the quality in 125 problems, and only decreases the quality in 49. The regression based strategy, *B10R*, improves the quality of 142 problems and decreases the quality in only 40. However, that is not an improvement over the classification based portfolios, *B5C* and *B10C*, respectively.

Those results mean that there is not one strategy perfect for all the criteria (number of problems solved and quality). With these results, the better strategy is *B10R* because it solves 13 problems more than *ET* and this strategy improves more plan qualities.

Table 5 follows the same structure than Table 4 but for the leave-one-domain-out evaluation mechanism: it shows for each portfolio strategy the number of problems solved, and the number of problems where the plan obtained has a quality higher and lower than the one obtained by LAMA-2011, respectively. Like in the results of the split evaluation, the best possible results are shown in the last column.

The best strategy with knowledge is *B10C*, which solves 12 problems more than the un-informed strategy, *ET*, and the same than than LAMA-2011. In fact, it is the only strategy with knowledge better than the un-informed strategy, *ET*. Comparing with the best portfolio of the competition, fdss-2, our technique obtained 29 problems more (fdss-2 solved 221 problems). These results mean that the planner combination is a good approximation for improving a single planner. But the results are worse than the split validation because the error in classification is much higher than in the split validation.

The bad result only affect significantly to two domains: Nomystery and Transport, where 3 and 7 problems are not solved by the portfolio strategies, respectively. In the other domains, there is a difference of only one or two problems.

As in the previous case, we also measured the quality of the plan obtained by the planners. In eight domains, the two evaluations (split and leave one domain out) obtain the same result (20 problems for domain). This domains are Barman, Elevators, Openstacks, Parking, Pegsol, Visitall and Woodworking. This group of domain suppose the 57.14 % the problems of the last competition and it is not a insignificant number, however this group is not enough. However this result is very significantly because the model do not have some any information about the domain and this task is difficult to realize.

The analysis of planning speed is not included because the created strategies focuses in obtaining the best plan for each problem in the maximumn time available (1800 seconds). If we would like to reduce the time where the best solution is obtained, we should create another strategies focusing in such objective.

	ET			BCE			B5C			B10C			B5R			B10R			Lama	BS	
	S	+	-	S	+	-	S	+	-	S	+	-	S	+	-	S	+	-	S	S	+
Barman	20	19	0	20	19	0	20	19	1	20	19	0	20	18	0	20	18	0	20	20	20
Elevators	20	16	2	20	16	1	17	14	4	20	18	0	18	15	3	20	18	1	20	20	20
Floortile	8	4	4	9	5	0	6	0	2	9	5	0	6	0	0	9	5	1	6	9	5
Nomystery	15	7	0	17	7	2	13	4	4	15	4	2	13	4	5	15	5	1	10	19	10
Openstacks	20	2	18	1	1	19	20	3	17	20	3	16	15	1	19	15	2	16	20	20	17
Parcprinter	20	0	20	20	11	0	20	5	12	20	11	0	20	5	12	20	11	0	20	20	11
Parking	12	3	16	20	4	12	20	2	12	20	4	12	20	2	13	20	4	15	20	20	9
Pegsol	20	0	8	20	0	2	20	0	2	20	0	2	20	0	2	20	0	2	20	20	0
Scanalyzer	18	2	14	17	8	4	17	4	6	17	4	7	18	4	6	17	4	6	20	20	13
Sokoban	17	5	10	19	5	1	18	1	7	19	4	1	18	2	6	19	s 5	1	19	19	6
Tibybot	16	5	9	18	5	4	19	6	3	17	4	7	15	3	7	16	3	7	16	20	13
Transport	20	9	11	13	12	7	16	8	7	13	10	7	13	8	9	13	8	10	19	20	18
Visitall	20	20	0	10	7	13	10	7	13	20	20	0	10	7	13	20	20	0	20	20	20
Woodworking	20	9	0	20	19	0	20	19	0	20	19	0	20	19	0	20	19	0	20	20	19
Total	246	101	112	224	119	65	236	92	90	**250**	**125**	**54**	226	88	95	244	122	60	250	267	181

Table 5: Comparison of the six portfolio strategies in leave-one-domain-out evaluation. Columns labeled with "S" show the numbers of problems solved in each domain. Columns labeled with "+" show the number of problems solved with plans of better quality to the ones reported by LAMA-2011. Columns labeled with "-" show the number of problems solved with worse quality than LAMA-2011.

Selection of Planners

The selection of the planners is performed automatically in the classification based portfolio strategies: for each problem, the classification based strategies decide a subset of planners to include in the portfolio. In Figure 2 we report the planners chosen by the B5C strategy in the split evaluation. In the x axis we show the domains used in the experiments and in the y axis we list the planners that the portfolio can use. The dot size indicates the number of times B5C selects a particular planner in a given domain. Given that we have 20 problems per domain, the maximum value is 20. In the case that B5C selected always the same planners for a given domain, there would be five points with the maximum size in the row corresponding with that domain.

The only planners that were never selected are ACOPLAN and ACOPLAN2. The most common selected planners are FD-AUTOTUNE-1 and RANDWARD. LAMA-2011 is not able to solve all the problems, and for some of the solved ones, it does not provide the best solution. Therefore, combining planners is a requirement to achieve the best results. Interestingly, B5C did not select LAMA-2011 for all the domains.

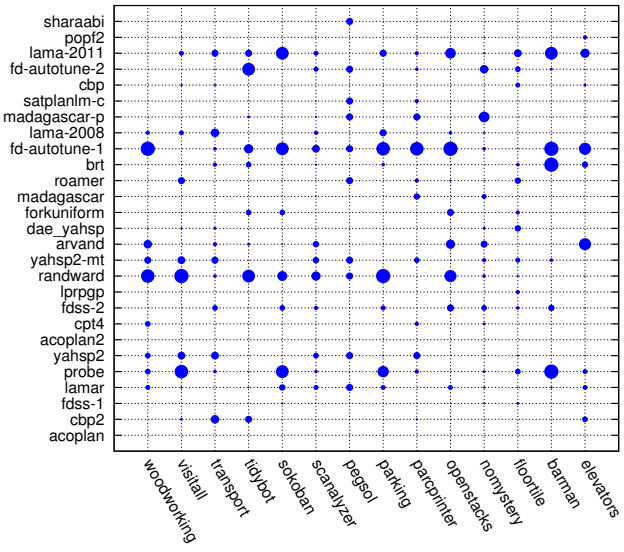

Figure 2: The planners selected by B5C for each domain

Related Work

Howe et al. (Howe et al. 2000) described one of the first portfolio planners. They implemented a system called BUS that runs only 6 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. They used only 5 features to represent the problems extracted from the PDDL description, while we characterize the problems with 47 features extracted from different sources, which has demonstrated that improve prediction capabilities (Cenamor, de la Rosa, and Fernández 2012). As in our case, the configuration of the portfolio can be different for different problems in the same domain.

Another portfolio (Gagliolo and Schmidhuber 2006) defines the same configuration for all the problems in the same domain. Each algorithm is run in parallel and dynamic

context-sensitive restart policies for SAT solvers are implemented. Another difference with our work is that they used SAT solver executions to learn the difficulty of the problems and split the time in between all SAT solvers, while we create models to predict planner performance.

Fast Downward planning system (Helmert 2006) includes the portfolios FD-Autotune and FD Stone Soup with several configurations. Each of these portfolios is a sequential portfolio planner that uses various heuristics and search algorithms. These algorithms are run consecutively for a total time of 1800 seconds. Each solver communicates to the following one the quality of the solutions found, and such value is used to improve the performance of the next solver. The same configuration is used for all the problems in the same domain. To learn the configuration, the authors used the results of different planning competitions, as we expect to do in the future.

Another portfolio, named PbP (Gerevini, Saetti, and Vallati 2009), learns a portfolio for a specific domain. PbP is not just a portfolio, because it also learns macro-actions for each domain and generate some portfolio configurations with them. Then, it runs the best three configurations in a round-robin strategy. This portfolio incorporate seven planners (Fast Downward, LPg-td, Macro-FF, Marvin, Metric-FF, SGPlan5, YAHSP). In a later version (PbP2 (Gerevini, Saetti, and Vallati 2011)) the authors introduced LAMA-2008 in the set of base planners. This portfolio won the learning track in the last IPC competition.

HYDRA (Xu, Hoos, and Leyton-Brown 2010) is a automated algorithm that combines portfolio-based algorithm selection with automatic algorithm configuration. They begin identifying a single configuration for a single problem, and spend all the time in this configuration. The configuration portfolios based only on a single highly parametrized SLS algorithm, SATenstein-LS (KhudaBukhsh et al. 2009). The main difference to our work is that these portfolios are focused on solving SAT problems.

ArvanHerd (Valenzano et al. 2012) is a satisfying parallel planner that won the last sequential multi-core track. This portfolio uses as base planners four configurations of the planner Arvand (Nakhost, Valenzano, and Xie 2011) an one configuration of LAMA-2008. In this case, the portfolio is fixed and it does not need to choose a subset of planners to run.

Conclusions and Future Work

In this work we have completed an analysis of the IPC-2011 result with a data mining methodology. With this analysis we built classification models for predicting whether a planner will success or not in a given problem, and regression models for predicting the time a planner will need to solve a given problem. We have introduced a set of elaborated features that come from the causal graphs and the domain transition graphs of the SAS+ formulation. The results have shown that these features are relevant for partially characterizing the complexity of the planning problems. Besides, these features are easy to compute, therefore they can be extracted in a pre-processing stage of a planning process. Then, the features are used to query a learned model for deciding the set of planners to use and the time they must be run.

We have defined a set of strategies to configure the portfolio and evaluated them with the problems of the IPC-2011. The results have shown that in 181 cases of 280, there exists at least a solution with better quality than that offered by LAMA-2011. This means that although LAMA-2011 is the planner that solves more problems, it is not the planner which provides the best plans. In addition, the ideal planner combination with all the planners in the competition solved more problems than the winner (17 problems).

With the analisys of the results, we have shown that the portfolios of planners are interesting in automated planning because there is not a best planner for all domains. The combination of the best planner in each domain is the perfect strategy, but this strategy is very difficult to obtain. The proposed option is learning what are the right planners to select, and we demonstrate that it is very close to that optimal solution. The results show that our strategies solved at least the 80% of the problems for previously unseen domains (leave one domain out evaluation). When affording new problems in known domains (split evaluation) the success raises over 92% in all the strategies.

In the future, we will try to learn better models for unknown domains to improve the performance of the portfolio. To achieve this goal, several strategies may be followed. A first one is to learn with more domains, so training data will cover a wider area of the domain space. A second one is to create new features that characterize the problems, as well as to apply feature selection approaches, to improve generalization capabilities. A third one is to perform a selection of the planners a priori, so we can discard planners that does not contribute to the global performance.

Acknowledgments

This work was partially supported by several Spanish projects: TIN2012-38079-C03-02, TIN2011-27652-C03-02 and TSI-090302-2011-6.

References

Backstrom, C., and Nebel, B. 1995. Complexity results for SAS+ planning. *Computational Intelligence* 11:625–655.

Briscoe, G., and Caelli, T. 1996. *A Compendium of Machine Learning: Symbolic Machine Learning*, volume 1. Ablex Pub.

Browne, M. 2000. Cross-validation methods. *Journal of Mathematical Psychology* 44(1):108–132.

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

Chapman, P.; Clinton, J.; Kerber, R.; Khabaza, T.; Reinartz, T.; Shearer, C.; and Wirth, R. 2000. Crisp-dm 1.0 step-by-step data mining guide. *http:///www.crip-dm.oprg/*.

Cristianini, N., and Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machines*. Cambridge University Press.

Dietterich, T. 2000. Ensemble methods in machine learning. *Multiple classifier systems* 1–15.

Frank, E., and Witten, I. H. 1998. Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learnin*.

Gagliolo, M., and Schmidhuber, J. 2006. Learning dynamic algorithm portfolios. *Annals of Mathematics and Artificial Intelligence, 47* 3(4):295–328.

Gerevini, A.; Saetti, A.; and 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.; and Vallati, M. 2011. Pbp2: Automatic configuration of a portfolio-based multi-planner. *The 2011 International Planning Competition*.

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

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

Howe, A.; Dahlman, E.; Hansen, C.; Scheetz, M.; and Von Mayrhauser, A. 2000. Exploiting competitive planner performance. *Recent Advances in AI Planning* 62–72.

KhudaBukhsh, A.; Xu, L.; Hoos, H.; and Leyton-Brown, K. 2009. Satenstein: Automatically building local search sat solvers from components. *Proc. of IJCAI-09* 517–524.

López, C. L. 2011. The seventh international planning competition documentation. Technical report, Universidad Carlos III de Madrid, Madrid, Spain. http://www.plg.inf.uc3m.es/ipc2011-deterministic/FrontPage/Software.

Nakhost, H.; Valenzano, R.; and Xie, F. 2011. Arvand: the art of random walks. *The 2011 International Planning Competition* 15.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Richter, S., and Westphal, M. 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *JAIR* 39:127–177.

Shevade, S.; Keerthi, S.; Bhattacharyya, C.; and Murthy, K. 2000. Improvements to the smo algorithm for svm regression. *Neural Networks, IEEE Transactions on* 11(5):1188–1193.

Valenzano, R.; Nakhost, H.; Muller, M.; Schaeffer, J.; and Sturtevant, N. 2012. Arvandherd: Parallel planning with a portfolio. In *ECAI 2012 - 20th European Conference on Artificial Intelligence.*, 786–791. IOS Press.

Wang, Y., and Witten, I. 1996. Induction of model trees for predicting continuous classes. Technical Report Working Paper 96/23, The University of Waikato.

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

Xu, L.; Hutter, F.; Hoos, H.; and Leyton-Brown, K. 2007. Satzilla-07: The design and analysis of an algorithm portfolio for SAT. In *Proceedings of the 13th CP Conference*.

Xu, L.; Hoos, H.; and 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)*, 210–216.