Mining IPC-2011 Results

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

The International Planning Competition (IPC) offers a wonderful scope to evaluate and compare different planning approaches and specific planner implementations in benchmark domains. In IPC 2011, the software generated for the competition permits to obtain a lot of data about the execution of all the planners in the different tracks in a simple way, which permits its recovery and use. In this work, we propose to analyze such data from a Data Mining (DM) perspective, including additional features which can be useful for the analysis. In such a way, we are able to construct models of the results obtained, allowing us to make additional analysis to the ones performed by the organizers. In this work, we report some initial analysis after constructing classification and regression models for the sequential satisficing and optimal track.

Introduction

Since 1998, the International Planning Competition (IPC) has offered the opportunity to researchers in Automated Planning to evaluate and compare their ideas about how to develop better and faster planners. Each competition produces a big amount of data specially from the execution of the planners in the different tracks, domains and problems. This fact is remarkable in the last two IPC, where the number of participants and the number of evaluated domains have increased considerably. Additionally in IPC 2011, the execution of a planner for a planning problem reported a total of 33 features, including runtime, number of solutions found, the moment where each solution was obtained, or the quality of those solutions. Such an amount of data opens a wide variety of analysis and studies from a Data Mining (DM) perspective. For instance, one would think whether it is possible to generate a model that predicts if a planner will be able to find a solution for a given problem and, if so, with what probability or how long it will take. The results of the prediction can help us to find some insights about the performance of planners or can be used to configure a portfolio of planners that takes into account the particular features of a planning problem.

In this work we perform an initial analysis of the IPC 2011 data. We follow a classical data mining methodology as it is introduced in the following section, which describes the DM process using a data workflow. The following sections explains the main steps deeply: the pre-process of the data, including feature generation, extraction and instance selection; then, the different data models generated, its evaluation and analysis. Later, some related works are summarized, while the last section describes our main conclusions and future research lines.

Data Mining Workflow of the IPC 2011 Results

Data Mining is a process of discovering implicit knowledge from data. Such process may contain different phases depending on the goals, data sources and tools. Figure 1 shows the complete process performed in this work. We have followed the phases described in the CRISP-DM methodology (Chapman et al. 2000): data understanding, data preparation, modeling, evaluation and deployment. [1]

In addition to previous phases, CRISP-DM starts with a business understanding phase, which is very related to business intelligence approaches, where the organization where the data mining process is going to be performed is analyzed to generate the data mining goals. In our case, we have defined the data mining goals as the creation of predictive models that predict, on the one hand, whether a planner will be able to solve a problem, and if so, what will be the time required to compute a plan. The first problem is a classification task, where the predicted attribute has only two possible values: $\{true, false\}$. 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). 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 improvements. 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 a portfolio-based planner.

Given those goals, the data source of the process is the SVN repository of the IPC 2011. From this repository we downloaded the domain and problem files in PDDL, and the competition results using the IPCReport tool. From the domain and problem files, we get the features of each instance

[1] In this work we do not study in deep methodological aspects, so any other methodology could be used for the description.

and compute additional features we think will serve for a better modeling. Once we have the features, the data is integrated, generating a first training set. This data suffers new transformations (feature generation and selection, and instance selection) depending on the requirements of the models to build. Afterwards, the models are learned using machine learning techniques and then evaluated using a suitable evaluation scheme for estimating the prediction capabilities of the models. In a real world scenario the methodology also includes a deployment phase, but we have not consider it for this work. The next sections describe in depth all these phases.

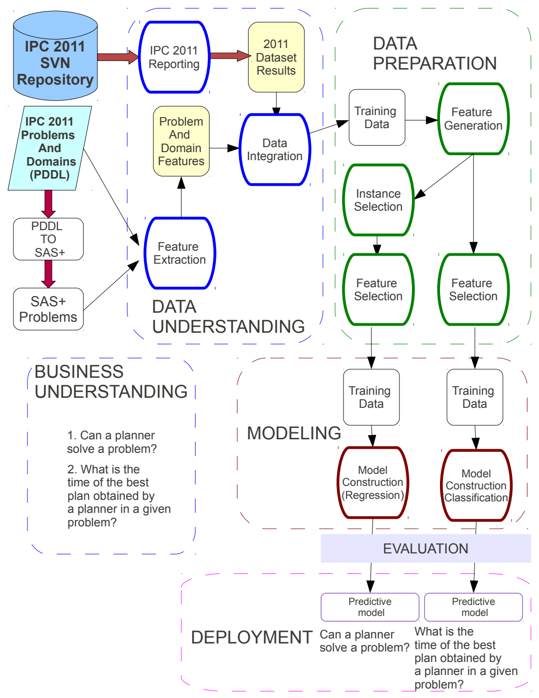

Figure 1: Data workflow of the mining process following CRISP-DM methodology

Data Understanding

This first step consists of recognizing the available sources of data and devising a good way to collect and integrate this data. For our analysis we have focused on the sequential satisficing track and the sequential optimal track of the IPC-2011. In the first one, planners should find a plan within the time bound. Since the solution does not need to be optimal, most planners develop the strategy of continuously reporting a solution that improves a previous one until the time runs out. In the sequential optimal track, planners should report a single optimal solution within the time bound.

For both tracks, the input data to the DM process comes from the competition results and from the domain and problem files. The competition results are in fairly suitable way for applying DM because each result for a planner execution, given a domain and a problem, corresponds to an example in terms of machine learning, i.e., a set of features and the given target attribute. However, this is not the case for PDDL files, therefore we need to extract some useful features from them. In the following sections we describe the data collected and generated for the mining process.

Features from IPC 2011 results

The IPC-2011 software (López 2011) contains several packages, which facilitates to test 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. The report is able to present 33 raw variables and 23 elaborated variables. The raw variables are the principal observations about the execution of planners for a given problem and domain. The elaborated variables are the ones derived from raw data, for example, a maximum of a set of values in the raw variables. In this work, we use some of the raw variables, which are described next:

1. *Planner*: the name of the planner.
2. *Domain*: the name of the domain.
3. *Problem*: a problem identifier (the problem number for a particular domain).
4. *Oktimesols*: For each validated plan found by the planner, a vector containing the time elapsed (in seconds) since the planner was started until such solution was found. This variable can either be empty (no solution was found), have a single value or an array of values. We will refer to this feature as the *time vector*.
5. *Values*: A vector containing the plan quality for the found solutions. The $i-th$ position of vector *OKtimesols* corresponds to the same plan as $i-th$ position of vector Values. We will refer to this feature as the *quality vector*

From the sequential satisficing track we got 7560 instances, corresponding to the execution of 27 planners in a total of 20 problems for 14 domains. From the the sequential optimal track we got 3360 instances corresponding to the execution of 12 planners in a total of 20 problems for 14 domains.

Features extracted from IPC 2011 domains and problems

The objective of this process is to generate a set of features that characterize a problem instance. In order to achieve a good characterization, we will use basic features that can be extracted from PDDL files and a set of elaborated features generated from the problem translation to the SAS+ formalism and its induced graphs, i.e., the causal graph and the domain transition graphs.

The basic features from a problem instance are:

1. *Objects:* Number of objects defined in the problem.
2. *Literals:* Number of instantiated predicates in the initial state.
3. *Goals:* Number of instantiated predicates that are true in the final state.

These basic features offer information about the complexity of the problems. In fact, most problem generators receives as input the number of each type object and the number of goals, so they can determine the instance size. However, these basic features and many others that can be extracted from the domain definition will not be sufficient for discriminating between instances of the same size. Indeed, any conceivable set of possible features from the domain, such as the number of actions, maximum predicate arity, maximum number of preconditions, in conjunction with the basic features from the problem, will serve to discriminate between problems of different domains or problem of the same domains with different size, but not between problems having the same object and goal distribution. We argue that additional information can be extracted from the graphs induced by the SAS+ formalism in order to partially recognize the differences between problems of similar size.

The SAS+ formalism (Backstrom and Nebel 1995; Helmert 2009) is an alternative representation for STRIPS. It considers a set of state variables, each one associated to a finite set of possible values. Actions have preconditions (i.e., required assignments of some variables to the action become applicable) and effects (i.e., the new values of the affected state variables). Using this formalism, a problem instance can be represented in a structured way using two types of graphs: (1) The ***causal graph*** (CG), which is a graph that captures the causal dependencies between the state variables of a given problem. (2) 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) pre-process to generate all graphs. We recall that in the causal graph, the *high-level* variables are the variables for which there is defined a 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.

Features from the Causal Graph: For the CG we classify the 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 or the standard deviation of particular elements of a graph. (4) high-level statistical, the same as before but only considering the high-level variables.

- **General Features**
  1. *NumberVariablesCG:* The number of variables of the causal graph.
  2. *High-LevelVariablesCG:* The number of high-level variables.
  3. *TotalEdgesCG:* The number of edges.
  4. *TotalWeightCG:* The sum of weights of the edges.
- **CG Ratios**
  1. *VERatio*: The ratio between the total number of variables and the total numbers of edges. This ratio shows the level of connection in the causal graph.
  2. *WERatio*: The ratio between the sum of the weights and the number of edges. This ratio shows the average weight for the edges.
  3. *WVRatio*: The ratio between the sum of the weights and the number of variables.
  4. *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**

  This statistical information is used to characterize the structure of the causal graph. We compute the average, the maximum and the standard deviation of the following values:
  1. *InputEdgeCG*: The number of incoming edges for each variable. Thus, we compute the average of input edges in the CG, the maximum number of input edges for a single variable and standard deviation of input edges, for knowing the variation between variables.
  2. *InputWeightCG*: The sum of the weights of the incoming edges for each variable. Thus, we compute three new features following the same computations as *InputEdgeCG*.
  3. *OutputEdgeCG:* The number of outgoing edges for each variable. Thus, we compute the average of output edges in the CG, the maximum number of output edges for a single variable and standard deviation of output edges.
  4. *OutputWeightCG:* The sum of the weights of the incoming edges for each variable. Thus, we compute three new features following the same computations as *OutputEdgeCG*.
- **Statistics of high-level Variables**

  This information tries to encode the structure for the variables involved in the problem goals. We compute the average, the maximum and the standard deviation for the following values:
  1. *InputEdgeHV*: The number of incoming edges for each high level variables. This value produces three new features following the same computation as *InputEdgeCG*.
  2. *InputWeightHV*: The edge weight sum of the incoming edges for each high level variables. This value produces three new features following the same computation as *InputWeightCG*.

Features from DTGs: Since the number of DTGs in a problem is variable, it is difficult to encode general attributes for each graph. Instead, we can summarize DTGs characteristics aggregating the relevant properties of all graphs. We compute general aggregated features and some statistics over all graphs.

- **General Aggregated Features**

1. *NumberVerticesDTG*: The sum of the number of vertices of all DTG.
2. *TotalEdgesDTG*: The sum of the number of edges of all DTGs.
3. *TotalWeightDTG*: The sum of the weights of the edges of all DTGs. The weight of An edge in DTG corresponds to the cost of applying the action that induced the edge.

- **Statistics of DTGs**

  Considering all DTGs, we compute the average, the maximum and the standard deviation of the following values:

1. *InputEdgeDTG*: The number of incoming edges for a vertex in a DTG.
2. *InputWeightDTG*: The sum of the weights of the incoming edges of all vertices.
3. *OutputEdgeDTG*: The number of outgoing edges for a vertex in a DTG.
4. *OutputWeightDTG*: The sum of the weights of the outgoing edges of all vertices.

Data Preparation

Data preparation may contain different phases, that can be summarized in four: data cleaning and transformation (like normalizing data), feature generation (generating new features from the available ones), feature selection (eliminate useless features) and instance selection (select a sub-subset of training instances from the total one).

The main tasks of the data cleaning are the management of missing values and the identification of outliers. For the first one, we have not performed any task since we have a strong confidence on the data. The IPCReport tool gives a complete and reliable data set and the feature generation process was computable for all problems used in the competition. For the second one, we decided to eliminate some outliers, mainly data from specific problems in some domains which generated data very far from the average values. Nevertheless initial evaluations demonstrated that results did not vary significantly.

Regarding feature generation, we generated a lot of derived features, as explained in previous sections. Although feature generation is an operation that is typically performed in this step, we performed it in the data collection and understanding phase due to implementation reasons, thus while we extracted the data from the PDDL and SAS+ representations, we computed also the derived features.

The rest of the data preparation processes are explained depending on the data-mining task:

- **Classification**: For this task we create a new attribute *class* representing whether a planner was able to solve a problem. This attribute is set to "yes" if there exists at least one value in the attribute of the *time* vector, otherwise it is set to "no". Both time and quality vectors are removed from the dataset since they are no longer of interest. Additionally, the domain name and problem number are treated as identifiers, therefore they will not be used for modeling. For this task, we did not perform any instance selection process, since data was very clean, and we have a manageable amount of instances. This task will be the same for the satisficing and optimal track.
- **Regression**: In this task we draw a distinction for IPC tracks. For the sequential satisficing track we try to predict three different values: *first-time*, representing the execution time elapsed when a planner found its first solution for the problem; *best-time* representing the execution time elapsed when a planner found its best solution; and *medium-time*, representing the median of the time vector. When predicting each of these values, the others are removed from the training set. The time and quality vectors are also removed. In addition, we have eliminated all the instances where a planner was not able to find a solution, i.e. where time and quality vectors are missing. In the sequential optimal track planners give one single time for finding the optimal solution, therefore the regression task consists of predicting the single value of the time vector. As before, we only consider instances where a solution was found.

Data Modeling and Evaluation

Although Figure 1 presents the data mining process as a cascade, data preparation, modeling and evaluation are typically performed many times iteratively until a satisficing solution is found. In addition, different models and learning algorithms are tested until the "best" one is obtained.

In this work, we have used several algorithms, decision trees (J48) (Quinlan 1993), 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. The implementation of these algorithms is provided by WEKA (Witten and Frank 2005), and they are used with the pre-defined parameters.

Evaluation Set-up

A question that arises when applying machine learning over any dataset is how to perform the evaluation process. Evaluating over the training set typically produces optimistic estimations of performance over future data, since it is easy that the generated models are over-fitted to the training set. Therefore, other evaluation mechanisms must be used. A classical one is *cross validation*, which is a procedure to estimate the performance of a previously learned model over data which has not been used to train the model. It consists on dividing the whole data set, D, in k slides, $D_1, \ldots, D_k$. Then, ten models are learned, $m_1, \ldots, m_k$, each of them using nine slices for training, and a tenth slices for test. Therefore, at the end of the process, we have k evaluations, $e_1, \ldots, e_k$, one for each model. The estimated performance e of the complete model is assumed to be the average of the ten evaluations. The standard deviation over such average usually gives information about how uniform the k slices are. The inductive hypothesis ensures that if the model has been learned over a large amount of representative data, then the cross validation will return an accurate estimation. However, is that true in our case? To answer this question, we divide it in two different ones:

1. is the estimation valid for new problems in the same domains seen in the IPC 2011?
2. is the estimation valid for new problems in domains different to the IPC 2011 ones?

The answer to the first question depends on whether the problems used for learning are a representative set of problems in such domain. Since the IPC goal is to evaluate planners in different situations, problems are selected to cover such objective. Therefore, we can expect that a classical cross-validation is a good estimation. This conclusion would not be strong enough if the distribution of problems (20 instances per domain in IPC-2011) only covers a small space of the possible problems.

In the case of the second question, that is equivalent to ask whether the set of domains is representative of the set of all the possible domains that can be defined in PDDL. To evaluate this issue, we need to modify the evaluation method to a leave-one-out approach. In classical supervised learning, a leave-one-out approach is equivalent to a cross validation where k is set to the size of the data set. This mechanism estimates how the model will behave in the next example received. In planning, we are interested in evaluating how the model will behave in a new domain. Therefore, we can apply a leave-one-out over domains, instead of on the whole dataset. Specifically, if we have data from k domains, we learn k models with the data from $k-1$ domains, and evaluate all them over the $k-th$ domain. The average of the k evaluations is the estimation of the performance of the models in future domains. We call this evaluation process as *leave-one-domain-out*. The metrics used are:

- **Classification Accuracy** is the percentage of instances in a data set that a classification model correctly classifies (the ratio between the correctly classified instances and the total, multiplied by 100). The classification accuracy can be measured over different data sets (training or test sets), or estimated through different processes, like split, cross-validation, or leave-one-out.
- **Relative absolute error** is the ratio between the absolute error of a prediction of a model (difference between the predicted value and the expected one) and the expected output. Since it is a relative value, it is more independent of specific class values than no relative measures. Opposite to classification where maximum accuracy is searched, regression is a minimization problem of the prediction error. It also can be computed over different data sets or estimated through different procedures.

Predicting Planner Success

The goal of this task is to predict whether a planner will solve a given problem. As described previously, the target variable belongs to the domain $\{true, false\}$, meaning whether the input planner was able to solve the input problem in the corresponding domain.

Empirical Results. Tables 1 and 2 show the classification accuracy and the standard deviation obtained in both evaluation processes for the sequential satisficing and sequential optimal track respectively. In each row, we show the results of the different classification algorithms: J48, IBK for different values of k, and SMO.

Dataset	Cross 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 1: Classification accuracy and standard deviation for predicting planner success in the sequential satisficing track with cross validation and leave one domain out evaluation

Dataset	Cross Validation	Leave one Domain Out
J48	**90.14(1.58)**	60.36 (23.69)
IBk -K 1	86.96(1.57)	60.36 (21.26)
IBk -K 3	87.81(1.81)	58.78 (21.66)
IBk -K 5	83.91(1.90)	60.86 (20.53)
SMO	79.96(2.30)	**67.41 (16.55)**

Table 2: Classification accuracy for predicting planner success in the sequential optimal track with cross validation and leave one domain out evaluation

Regarding the sequential satisficing track, the cross-validation results show that the best result is obtained using decision trees (J48), but IBK obtained fairly similar accuracy. These results reveals that we are able to achieve good classification accuracy in the cases where we are predicting planner success within already seen domains. However, the results for the leave-one-domain-out evaluation worsen considerably, showing that it is very difficult to generalize in the classification task for new unseen domains. Besides, both evaluation schemes show that we are able to create models that are better than a default or random classifier, which would obtain an accuracy of 50.7%. This percentage corresponds to the ratio of successful executions in the track. Concerning the sequential optimal track, algorithms performed similar than in the seq-sat track. As before, the leave-one-domain-out evaluation got worse results than cross validation, though the best result (67.4%) was higher than in seq-sat results.

We also have performed a feature selection process prior the generation of the models. Nevertheless, the default parameters produce an aggressive process. As a result, only one feature was left after the process (i.e., the planner). With only this feature, generalization is not possible, and the classification result was very poor (72.06 ± 1.52) independently of the algorithm). We have not applied other configuration or algorithms for feature selection. We postpone this study to future research.

We performed an additional evaluation to see if we can predict the performance of some planners better than others. For achieving this, we made a separate dataset for each planner (a total of 27). Then, we built the models with J48 (decision trees) and evaluated them using cross validation. Table 3 shows the results for the best, the worst and the average predicted performance. We also include the accuracy for the winner, LAMA. As we can see there is considerable gap in the classification accuracy. Another relevant observation is that it is hard to predict the performance of a planner based on an algorithm-portfolio, which internally could behave as different planners. This is the case of the Fd-autotune-2 which produced the worst model.

planners		accuracy
Minimum	Fd-autotune2	78,2
Maximum	Acoplan, Acoplan2	97,5
Average	–	88,5 ± 5,3
Track Winner	Lama-2011	81,4

Table 3: Different classification accuracies achieved with individual models

Semantic Analysis. Here we give a few examples of the knowledge that we could extract from the models learned. Specifically, we use the PART algorithm implemented in WEKA to obtain a set of decision rules that predict whether a planner will succeed or not. This algorithm firstly creates a decision tree, from which it generates the set of decision rules, which are then pruned (simplified). It also includes a parameter (minNumObj), which sets the minimum number of instances per rule allowed. This is, therefore, a prune parameter. If it is low, it permits models with a large number of rules (maybe sub-optimal due to over-fitting to the training data), which hence, are more complex to understand for a human. However, while this value grows, the models generated contains less rules (maybe sub-optimal due to the lack of expressiveness). In machine learning, balancing expressiveness capability and over-fitting risk is a main issue. Table 4 shows the importance of a correct balance. It shows the prediction capability of the models generated for different pruning values over the training set and estimated by the cross validation. This simple evaluation demonstrates that reducing pruning capabilities increases expressiveness (the generated model has more rules), but it over-fits to training data (difference between training success and cross-validation estimation is larger). A good balance, in this case, is a pruning value of 10, which maintains high accuracy (although significantly worse than the others) without over-fitting, with only 120 rules.

What is the looking of the rule sets generated? Are they really informative? The following is the rule at generated for the pruning parameter set to 1000, which is composed of only four rules.

```
Rule 1: NumberVerticesDTG > 168 AND
  AND STDInputEdgeHV <= 42.9906
  AND AVGInputEdgeDTG > 1.84576
  AND TotalEdgesCG > 699: false  (2160.0/926.0)

Rule 2: NumberVerticesDTG > 168 AND
AVGInputEdgesDTG > 1.84576: true (2349.0/1024.0)

Rule 3: STDInputEdgeHV <= 55.8043: false (1566.0/560.0)

Rule 4: true (1485.0/459.0)
```

Pruning parameter	training success	cross-validation success	number of rules
1	94.9	88.8 ± 1.15	287
2 (default)	94.4	88.6 ± 0.83	233
10	89.9	85.8 ± 1.41	120
100	78.5	75.7 ± 1.44	31
1000	60.72	60.3 ± 1.55	4

Table 4: Different results of PART depending on the pruning parameter (minimum number of instances per leaf) and the evaluation mechanism (over training data or estimated with cross-validation procedure.

The way to read the rules is the following. First, we check the conditions of the first rule: ($NumberVerticesDTG >$ 168 AND $STDInputEdgeHV <=$ 42.9906 AND $AVGInputEdgeDTG >$ 1.84576 AND $TotalEdgesCG >$ 699). If the condition is true, the output is false (i.e. the problem can not be solved). The values $(2160.0/926.0)$ indicates the number of instances that satisfy the condition and that belongs to the class 'false' and 'true' respectively. Therefore, those values provide an idea of the coverage and success of the rule. If an instance does not satisfy Rule 1 condition, Rule 2 is checked, and so on. The last rule returns a default value in case the input instance does not satisfy any previous rule.

We want to highlight that this rule set is so reduced that it does not ask for many of the input features that describe the instances. It is so reduced that it does not ask for the planner. However, it already gives information about the problems, and it seems that to have a number of variables in the DTG higher or lower than 168 is an important feature of the problems, since that condition appears in the the initial rules. In fact, $NumberVerticesDTG$ is a feature that appears in all the decision trees that PART construct to generate the rule sets, independently of the pruning parameter. For instance, when the minimum number of instances per leaf is 2, the first question that the decision tree generated makes is whether $NumberVerticesDTG$ is larger than 106. Then, it asks for the planner.

For higher expressiveness, we need to reduce pruning capabilities, and we can try to understand what are the features that characterize whether a problem will be solved or not. For instance, this is the rule set that characterize the problems that LAMA is not able to solve (for a pruning parameter set to 2):

```
MAXInputWeightDTG > 3: true (166.0/1.0)
HVRatio > 0.970588 AND
  STDInputEdgeCG > 0.026307:true (27.0)
MAXInputWeightHV > 70 AND
  AVGInputWeightHV <= 11520: true (37.0/1.0)
WVRatio > 16896: false (9.0)
MAXInputWeightCG <= 6480: true (20.0/1.0)
goals <= 14: false (4.0)
otherwise: false

literals <= 9801 AND
MAXInputWeightHV > 60: true (164.0/2.0)
```

```
NumberVariablesCG > 58 AND VERatio > 0.00575 AND
MAXInputEdgeHV <= 8: true (51.0)

Goals <= 50 AND
NumberVerticesDTG > 130 AND NumberVariablesCG > 11 AND
MAXInputEdgeDTG <= 1210: false (22.0)

MAXInputEdgeDTG <= 1397: true (36.0)

: false (7.0/1.0)
```

Can this rule set help researches to focus the development or improvement of their planners? That is something that only them can answer but, at least, these rules can give some clues, since they characterize the solved and the unsolved problems. A similar study could be easily performed for any of the planners of the competition, although we omit them due to lack of space.

Predicting Execution Time

Once we know whether the input planner will be able to solve the input problem (using the predictive model learned in the previous phase), the goal of this task is to predict the time a planner needs to find the solutions.

Empirical Results. As explained before, in the sequential satisfying track we have created models for predicting the time invested in finding the first, the median and the best solution. The results obtained for this track are shown in Table 5. We follow the same evaluation scheme of the classification case. The table shows the relative absolute error and standard deviation after the validation process. As in the classification case, results worsen when estimations are performed with the leave-one-domain-out scheme. Besides, IBK with different values of K had less degradation of the estimated errors, showing that this technique could be better when making predictions on new domains.

Regarding the sequential optimal track, if planners report a solution, it should be optimal. Therefore, it only make sense to create a model to predict a time for finding the solution, which is obviously the best one. Table 6 shows the results for both evaluation schemes. Each value corresponds to the relative absolute error and the standard deviation after the evaluation process.

Dataset	Cross Validation	Leave Domain Out
M5Rules	67.08(7.63)	213.87 (231.95)
IBk -K 1	**59.74(8.37)**	141.54 (47.40)
IBk -K 3	59.99(6.32)	**123.37 (11.26)**
IBk -K 5	63.59(6.38)	127.21 (10.96)
SMOreg	66.84(5.71)	15151.04 (54178.83)

Table 6: Relative absolute error (% percentage) and standard deviation of predicting the time planners will need to find a solution in the sequential optimal track

Semantic Analysis. Semantic analysis of regression models is more complex than for classification, because regression models include mathematical models difficult to interpret. In our case, we will analyze a few of them. For this study, we use the algorithm M5Rules, which generate regression rules. Table 7 shows the results of an experimental process equivalent to the one described in Table 4.

Pruning parameter	Training Error	Cross-validation Error	Number of Rules
1	71.26	71.26 ± 6.82	5
4 (default)	71.26	73.38 ± 6.82	5
10	66.68	71.63 ± 3.08	11
100	70.42	71.35 ± 2.30	11
1000	83.86	77.33 ± 2.02	4

Table 7: Different results of M5Rules depending on the pruning parameter and the evaluation mechanism (over training data or estimated with cross-validation procedure.

As in the case of classification, we can see that a value of 10 is a good value for the pruning parameter. However, in this case, the number of rules generated is not so different for different values, as well as the prediction capabilities. Again, we analyze the looking of the rule set generated for LAMA planner In this case, M5Rules generates only 2 rules, each of them with its linear regression model. From a semantic point of view, we can think that the regression models generated could give an idea about the importance of the features in the prediction model. For instance, in this case the output value changes a lot with different values of the $VERatio$ and $MAXInputWeigthDTG$ features (in rule 1), and with $AVGInputEdgeHV$, $STDInputEdgeCG$ and $STDOutputEdgeCG$ (in Rule 2), showing that, for LAMA, the output time is very sensitive to small differences in those values. The relative absolute error of that rule set is 78.74%.

```
Rule: 1
IF STDInputWeigthCG <= 13305.8
MAXInputWeigthDTG <= 4.5
WERatio > 5.52
THEN bestTime =
0.17 * objects
- 0.0764 * goals - 0.0002 * TotalWeigthCG
- 52.7647 * VERatio - 0.0099 * WERatio
- 0.0676 * WVRatio  + 0.7708 * AVGInputEgdeHV
- 0.4982 * STDOutputEdgeCG + 0.0043 * STDInputWeigthCG
- 0.0002 * TotalWeigthDTG - 0.0226 * NumberVerticesDTG
+ 0.0015 * TotalEdgesDTG + 0.0703 * STDInputEdgeDTG
+ 0.0314 * MAXOutputEdgeDTG - 8.4912 * MAXInputWeigthDTG
+ 178.6288 [79/44.805%]

Rule: 2
bestTime = 0.0784 * literals  + 0.0003 * TotalWeightCG
+ 6.8833 * AVGInputEdgeHV - 0.0053 * STDInputEdgeHV
+ 11.8598 * STDInputEdgeCG - 11.4135 * STDOutputEdgeCG
+ 193.2316 [171/85.323%]
```

These analysis are only small examples of how powerful data-mining studies can be for supporting decisions in solving planning problems. For instance, one classical decision in heuristic search based planners is, once a solution is found, to decide whether to wait for a new solution is interesting or not. To make this decision, to predict the required

Dataset	Cross Validation			Leave Domain Out		
	First Time	Medium Time	Best Time	First Time	Medium Time	Best Time
M5Rules	73.81(4.78)	74.02(3.90)	73.66(3.61)	17204.81(60518.16)	1492.24(2798.89)	985.64(2200.93)
IBk -K 1	59.84(5.15)	65.25(5.28)	67.57(4.07)	87.94(30.76)	91.12(29.39)	93.66(23.38)
IBk -K 3	**55.05(3.72)**	**60.02(4.00)**	**62.98(3.12)**	**79.31(28.27)**	89.87(31.70)	85.96(22.26)
IBk -K 5	56.61(3.66)	60.93(3.51)	64.39(3.00)	92.12(29.73)	**89.70(26.57)**	**85.57(19.21)**
SMOreg	60.18(4.06)	64.08(3.65)	69.50(2.87)	835.17(2264.22)	184.10(165.75)	907.32(2620.74)

Table 5: Relative absolute error (% percentage) and standard deviation of predicting the time planners invested in finding the first, median and best solution in the sequential satisficing track.

time to obtain the best plan is required. We can predict that value for LAMA with a mean absolute error of only 217 seconds.

Related Work

The construction of models to predict the performance of planners is not a novel idea. Roberts and Howe (2009) showed that model learned from planners' performance on known benchmarks up to 2008 get high accuracy when predicting whether a planner will succeed or not. They use 19 features extracted from the domain and problem definition. They also proposed to use some features from the causal graph, but they did not find them relevant for the classification task. The main difference with our approach is that we include features also from the domain transition graphs and most of our features come from the ground instantiation of the problem. Additionally, results from both works are not comparable because they do not have common planners or instance sets. Besides, they found that domain features are the more relevant ones, and models are indeed basing their planner success prediction on implicitly predicting the domain. This is a good insight regarding the available data; however, it does not seem a general rule, since one can imagine large planning problems for which any planner will fail to solve it. We think that models may exploit instance features in order to determine the relative hardness between instances of the same domain. To achieve this goal, further investigation is needed and performance data should be collected from instance sets with more diversity than the one that could be derived in 20 instance per domain used in the IPC.

In a more general scope, the prediction of a solver performance is a research topic of interest since it is one of the techniques for creating algorithm portfolios (Xu et al. 2007; Gagliolo and Schmidhuber 2006). The idea consists of learning empirical hardness models based of instance features that can be computed efficiently. Then, whenever the portfolio tries to solve a new instance, the learned models predict the set of solvers that are likely to solve the instance, so the available time is scheduled between the selected solvers using a criteria based on the prediction confidence. This approach can make a per-instance decision of which portfolio configurations are likely to perform the best. However, the successful portfolios in automated planning (Gerevini, Saetti, and Vallati 2009; Fawcett et al. 2011) are only able to use the configuration of the best average performance in a set of training problems. Therefore, given a new set of instances, these portfolios tries to solve them using a fixed configuration (i.e., the best over past benchmarks) or a per-domain configuration, if there is an available example set of a particular domain. This per-domain decision for configuring portfolios is the approach followed by the cited planners in the learning track of the IPC. We argue that one can use the models presented in this work (or other similar ones) to create a per-instance configurable planning portfolio.

Conclusions and Future Work

We have presented an analysis of the IPC-2011 results following a data mining methodology. With this analysis we have given some insights about the performance of planners, in addition to the general results reported by the IPC organizers. We have built classification models for predicting whether a planner will succeed or not in a given problem, and regression models for predicting the time a planner will need to solve the problem.

We have presented the *leave-one-domain-out* evaluation scheme. This kind of evaluation is an alternative to the standard cross-validation if one want to estimate how good the learned models are, in the situations where problems belong to an unknown domain. As we have seen in the results, we can get good classification accuracy when we are dealing with problems of known domains, but it seems that this does not hold in unknown domains. We expect that if we reproduce the same experiment with more domains, the models would generalize better.

We have introduced a set of elaborated features that come from the causal graphs and the domain transition graphs. The results have shown that these features are relevant for partially characterizing the complexity of planning problems. Besides, these features are easy to compute, therefore they can be extracted in a pre-processing stage of a planning process, and then used to query a learned model for deciding the set of planners and the set of times in an instance-based configurable portfolio. In the near future we plan to continue our research in this direction.

Acknowledgments

This work was supported by several Spanish projects: TIN2008-06701-C03-03, TIN2011-27652-C03-02, TIN2010-08861-E and TRA2009_0080.

References

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

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.

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

Fawcett, C.; Helmert, M.; Hoos, H.; Karpas, E.; Roger, G.; and Seipp, J. 2011. Fd-autotune: Domain-specific configuration using fast downward. In *Booklet of the 7th International Planning Competition*.

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)*.

Helmert, M. 2006. The fast downward planning system. *JAIR* 26:191–246.

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

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.

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.

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

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*.