Universidad
Carlos III de Madrid

TESIS DOCTORAL

Creating Planning Portfolios with Predictive Models

Autor:
Isabel Rosario Cenamor Guijarro

Director/es:
Dr. D. Tomás Eduardo de la Rosa Turbides y
Dr. D. Fernando Fernández Rebollo

Departamento de Informática. Escuela Politécnica Superior

Leganés, 23 Marzo 2017

TESIS DOCTORAL

CREATING PLANNING PORTFOLIOS WITH PREDICTIVE MODELS

Autor: Isabel Rosario Cenamor Guijarro

Directores: Dr. D. Tomás Eduardo de la Rosa Turbides y
Dr. D. Fernando Fernández Rebollo

Tribunal Calificador		Firma
Presidente:	Araceli Sanchis de Miguel	....
Vocal:	Álvaro Torralba Arias de Reyna	....
Secretario:	Alessandro Saetti	....

Calificación: ....

Leganés, 23 de Marzo de 2017

"Sometimes it is the people who no one imagines anything of who do the things that no one can imagine."
– ***Alan Turing***

To Carlos.

Agradecimientos

Firstly, I would like to express my sincere gratitude to my supervisors Tomás and Fernando, who gave me continuous support, guidance, motivation and their knowledge throughout this thesis. Thank you very much for your patience and understanding, and for the trust you placed in me. Your guidance helped me throughout the research and writing of this thesis. I could not have imagined having a better mentors for my Ph.D.

I would also like to thank my fellow lab mates Álvaro, Vidal, Pulido, José Carlos, Alberto, Javi who helped to create the nice workplace where I like to go, even though not all of all the moments are always good moments, but it is not possible to find a better without them. Over the years, we had a lot of great moments and not just at the university, in bars, on holidays, excursions amongst others. I hope these moments can be repeated when this period of my life comes to an end, because you guys are not only mates, you are trusted friends too.

It is impossible to forget the rest of the PLG group: Raquel, Daniel, Carlos, Ángel, Susana, Neli are just some who spring to mind. I think it is the best group ever in which do a Ph.D. Also, I want to thank other university people that I met during this time, especially those from the gym. It was one of the best places to go, when I was mentally blocked: get to the gym and work out.

My sincere thanks also goes to Lee, Mauro and Lukas who gave me the opportunity to join their team as an intern, and who gave me access to their research group for almost four months.

I thank my friends: Leti, Jesús, Santi, Vero, and Montoya. I will never forget all the chats, happy and beautiful moments that I continuously shared with my friends. They are fundamental in my life and especially when they supported me during these stressful and difficult moments. I would also thank my university friends Alex, Oscar, Alberto, Irene and Javi, with whom I spent lots of time at the university.

Y sobre todo quiero dar las gracias a mis padres y a mi hermano por el apoyo que me han dado en esta etapa de mi vida y en todas ellas. Siempre me han ofrecido consejo, amor y ánimo cuando lo he necesitado. En último lugar, pero no por ello menos importante, gracias a Carlos por todo su amor.

A todos vosotros, muchas gracias.

Abstract

Sequential planning portfolios are very powerful in exploiting the complementary strength of different automated planners: for each planning task there are one or more base planners that obtain the best solution. Therefore, the main challenge when building a planning portfolio is to ensure that a suitable planner be chosen and that it gets enough planning time. To solve this problem we need firstly to define three elements. The first is the settings or planning conditions: time, memory, or other constraints. The second one is the set of base planners. And finally, a benchmark that provides us with knowledge on how the base planners will behave under the given settings, following some kind of inductive process. Ideally, if the previous elements are correctly defined, when a new planning task arrives, an oracle will be able to tell which base planner to run and for how long. In practice, since no oracle exists, the challenge to choose a sub-set of base planners, is assigning them a running time and deciding the order in which they are run to optimize a planning metric under the predefined settings. Many state-of-the-art portfolios might never achieve an optimal performance because they do not select different planners for the different planning tasks. In addition, these static techniques typically assign a fixed running time to the selected set of planners, independently of the task. besides, the old-fashioned dynamic portfolios present a poor characterization of the planning task and do not have enough knowledge to predict an accurate portfolio configuration in many cases. The aforementioned drawbacks are intensified by the fact that there is an increasing number of planners available to choose from, although many of them are designed following similar approaches, so they are expected to behave similarly.

This dissertation is built on two main hypotheses. Firstly that the space of the base planners can be reduced just by selecting a subset of diverse or complementary planners; e.g. that there is a minimal set of planners that ensure that the optimal portfolio can be computed. Secondly, that planning tasks can be characterized, and that the difficulty in solving them can be modelled as a function of these features. To evaluate the first hypothesis, we analyze different metrics that could be used to filter the initial set of base planners. Classical metrics such as coverage, quality or execution time have been chosen by different portfolios in the past. We demonstrate that these selection methods may reduce the diversity of the portfolios, and propose an alternative method based on the Pareto dominance. We then carry out a profound analysis on previous planning task characterizations and show

how we could exploit them in current planning paradigms. A group of very informative features are proposed to improve the current feature definition of the planning tasks. These features have enough knowledge to differentiate planning tasks with similar “a priori” complexity. In this thesis we demonstrate that the implicit knowledge can be exploited in the construction of predictive models. These models estimate whether a base planner will be able to solve a given problem and, if so, how long it will take. Nevertheless, the predictive models are not perfect and sometimes provide wrong (or inaccurate) predictions. To solve this kind of problems, we propose different portfolio strategies to combine the number of selected base planners and their times. These strategies take into account the predefined settings and the knowledge learned in previous phases.

In conclusion, this thesis sets out a profound analysis of three different mechanisms or steps to create planning portfolios with predictive models, including new proposals for developing: planner filtering, planning task featuring, learning predictive models and portfolio construction strategies. One of the proposed portfolios was the winner of the Sequential Satisficing Track of the International Planning Competition held in 2014.

Resumen

Los portfolios de planificadores tienen un gran potencial ya que pueden aprovecharse de los diferentes planificadores automáticos, consiguiendo mejorar el rendimiento de un único planificador. Sin embargo, la creación de un portfolio no es una tarea sencilla, ya que para poder crear uno lo suficientemente bueno, hay que tratar tres problemas fundamentales. El primero de ellos es encontrar qué planificadores hay que seleccionar como componentes del mismo. La segunda es el tiempo que hay que asignar a cada planificador y, la última y no menos importante el orden en el que se tienen que ejecutar. Actualmente en el estado del arte, estas configuraciones, se realizan a partir de los resultados obtenidos por los planificadores en una fase previa de entrenamiento con un conjunto de problemas y restricciones prefijado (tiempo, memoria, etc), consiguiendo una configuración específica tratando de optimizar una métrica. Idealmente, la mejor configuración posible consiste en asignar el tiempo suficiente al mejor planificador para cada tarea de planificación. Sin embargo, esta configuración no siempre es posible, y hay que recurrir a otras aproximaciones como asignar un tiempo fijo a una selección de planificadores. Ésta no es la única simplificación utilizada, existen otras técnicas más cercanas a la óptima, en las cuales se selecciona un planificador o varios en función de la tarea a resolver. Sin embargo, estos sistemas, denominados dinámicos, incluyen una escasa caracterización de las tareas de planificación.

En esta tesis se parte de dos hipótesis. La primera de ellas es que existe un conjunto reducido de planificadores que maximiza la diversidad. La segunda de ellas consiste en la posibilidad de crear un conjunto de descriptivos lo suficientemente bueno para caracterizar la tarea de planificación. La caracterización de las tareas de planificación puede estar basada en sus distintas representaciones, así como en sus paradigmas. La primera tarea es seleccionar un conjunto de planificadores; realizando un análisis basado en las métricas clásicas de planificación, como son problemas resueltos, calidad y tiempo para seleccionar un subconjunto de planificadores. Adicionalmente, proponemos como alternativa a estas métricas, una técnica multiobjetivo. Este criterio está basado en la dominancia de Pareto combinando las métricas de tiempo y calidad. Continuando con nuestras hipótesis es necesario crear un conjunto de características bien informado para la tarea de planificación. Estas características deben ser capaces de diferenciar adecuadamente por problema y para ello será necesario basarse en los distintos paradigmas de la planificación automática. Este grupo de características tienen que ser útiles para crear modelos predictivos. Estos modelos podrán darnos además de una selección de planificadores, una aproximación del tiempo asignado a cada componente y el orden de los mismos. Adicionalmente se presentarán una serie de estrategias para explotar el conocimiento obtenido con los modelos predictivos.

En conclusión, se plantea y desarrolla un sistema para configurar porfolios de planificadores usando modelos predictivos en tres fases distintas. Una instanciación de este sistema fue el ganador de la competición internacional de planificación en el área de satisfacibilidad en el año 2014.

Contents

List of Figures

List of Tables

Glossary

ADL	Action Description Language
AI	Artificial Intelligence
AP	Automated Planning
ASP	Answer Set Programming
AUROC	Area Under the ROC Curve
Avg	Average
BN	Best Number
C	Coverage
CFS	Correlation based Feature Selection
CG	Causal Graph
CNF	Conjunctive Normal Form
CPU	Central Processing Unit
CSP	Constraints satisfiability problems
DCK	Domain-specific Control Knowledge
Def	Default
DTG	Domain Transition Graph
EPM	Empirical Performance Modeling
ET	Equal Time
FB	Fact Balance
FD	Fast Downward
FDSS	Fast Downward Stone Soup
FF	Fast Forward
FPR	False Positive Rate
GB	Gigabyte
GBFS	Greedy Best-First Search
GPU	Graphics Processing Unit
GR	Gain Ratio
HSP	Heuristic Search Planner
HV	High Level Variables
IBaCoP	Instance Based Configured Portfolio
IBaCoP-BNE	Instance Based Configured Portfolio - Best *N* Estimation
IPC	International Planning Competition
LISP	LISt Processing
LS	Local Search
Max	Maximum
Min	Minimum
MIP	Mixed-Integer Programming
ML	Machine Learning
nT	non Temporal
OET	Overall Equal Time
PCA	Principal Component Analysis
PDDL	Planning Domain Definition Language
Q	Quality
RAE	Relative Absolute Error
RAM	Random Access Memory
Rand	Random
RHW	Richter, Helmert and Westphal
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
RP	Relaxed Plan
SAT	Boolean satisfiability problem
SBS	Single Best Solver
SD	Standard Deviation
Sel	Selection
seq-sat	Sequential Satisficing

GLOSSARY

STRIPS	Stanford Research Institute Problem Solver
T	Time
TFD	Temporal Fast Downward
Tp	Temporal
TPR	True Positive Rate
VAL	The Automatic Plan Validator
Var	Variance
VBS	Virtual Best Solver
VMR	Variance-to-mean ratio

1 Introduction

This chapter provides a general overview of this thesis, including an overall idea of Automated Planning and Machine Learning. In addition, this chapter provides the objectives of the thesis, its main contributions and, finally, a brief outline of the rest of the document.

1.1 Overview

Planning is the process of deciding what to do in the future. More specifically: "Planning is the reasoning side of acting. It is an abstract, explicit deliberation process that chooses and organizes actions by anticipating their expected outcomes" (Ghallab et al., 2004). The objective of this deliberation is to achieve some predefined objectives in the best possible way. Automated planning is the area of Artificial Intelligence (AI) that studies this deliberation process from the computational point of view.

Formally, Automated Planning (AP) is the area of Artificial Intelligence (AI) that studies the explicit deliberation processes that choose and organize actions in steps to set out a plan (Ghallab et al., 2004, Russell and Norvig, 1995). Automated planning tasks are made up of a domain and a problem definition. The domain is defined as a set of permitted actions and rules that allow the state of the world to be defined; and the problem is a set of instantiated predicates that define an initial state and a goal state and that takes into account the rules defined in the domain model. An example of a planning task is the transportation of packages between two different locations. In this situation, the plan depends on the constraints of the transport system and its fuel consumption, the knowledge you have as regards the problem, etc. Depending on how the planning task is described, the planning system could be different. This work is centred on domain-independent planning, but there are others such as domain-specific planning and domain-configurable planning. *Domain-Independent Planning* uses generic representations and techniques to carry out the planning tasks. The planning engine is general enough to work in a planning domain that satisfies any set of simplifying assumptions. These techniques are able to change the domain easily. However, this mechanism does not guarantee efficiency in all possible planning problems. The second type, *Domain-specific Planning*, uses specific representations and techniques adapted to each planning problem or each planning domain. In this case, the planning engine takes advantage of the specific knowledge of the domain, and this type of planning usually obtains better results than domain-independent approaches but is less effectiveness in the rest of planning domains. The last one, *Domain-configurable planning*, uses specific representations and techniques only useful for a specific problem or domain, these modifications are sometimes included in the planner. These planning systems, which are tailor-made for use in a given planning domain (or problem), are unlikely to work in other domains unless major modifications of the planning system are carried out. In this thesis, we will focus in domain-independent planning, and we propose the reuse previously developed planning systems to carry out new planning tasks.

All previous types of planning are concerned with choosing and organizing actions for changing the state of a system from an initial situation to another one in which several goals are satisfied. Formally, AP requires a conceptual model in which the elements of the task to be carried out are described. This conceptual model is interesting because it is good for explaining basic concepts, proving semantic properties and clarifying assumptions (Ghallab et al., 2004). In this context, in 1998, the planning community set up a competition, the International Planning Competition (IPC) , to establish a common language and framework for comparing automated planners. Until 2008, the IPC was a biennial event. After that, it was held every three years (2011 and 2014), however, there are no more planned at the moment. This event remains a keystone in the planning community. The IPC generates a global standard in which any planning system may be compared with others. In each competition, different planning systems compete in different tracks with different metrics (or planning goals), and only one planner is declared the winner of each track. However, one of the main invariants of the competition is that there is no single planner which is always the best (or at least the same) for every domain or 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 the AP community has generated a set of single planners that are better than all of the others in specific situations. For this reason, discarding these best planners seems meaningless. This situation leads us to one of the bases of this thesis: the use of planning portfolios could improve the performance of every single planner.

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 AI. For instance, in Machine Learning (ML), meta-classifiers use different base classifiers 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 (Malitsky et al., 2013, Xu et al., 2010, 2008). For example, the SAT Competition has a special track on systems that included more than two solvers (Open track). In the automated planning community, planner portfolios have also been subject to great deal of interest. In IPCs from 2006 to 2014, portfolio approaches obtained good results in almost every track in which they took part.

However, although the use of portfolios has become the norm in the community, there is still no agreement as to what a planning portfolio is (Vallati et al., 2015). In this work, we assume that a portfolio of planners is a set of planning engines selected and scheduled according to an exploited automatic selection strategy. This selection strategy is what generates a specific portfolio configuration, whose goal is to maximize the performance metrics. 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 for configuring 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 behaviour, 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 large, and a portfolio with all available planners is an impractical idea. Nevertheless, if a planner is discarded in an initial phase, it might not be useful in future planning tasks since, otherwise, its planning capabilities will decrease. Therefore, one hypothesis of this thesis is that there is a small group of planners with enough diversity to be able to solve the same planning tasks as using all available planners. A planner filtering is an option to select the minimum number of planners to ensure that the best performance for each evaluated domain is achieved, or even for every problem in each domain. Obviously, good results in current domains do not ensure good results in new domains but, as will be demonstrated in this thesis, it is a good estimator. We evaluate filtering strategies with classic metrics (like time, quality and coverage) and establish that they are not good enough in some cases. In this sense, we propose a Pareto efficiency-based approach (Censor, 1977) to reduce the number of planners that we consider eligible for a planning portfolio. However, we will show that with this mechanism, the first of the aforementioned questions (which is the minimal set of base planners to use) can only be answered partially since the number of candidate planners might still be considerable.

Ideally, the best solution to the portfolio configuration problem is to have an oracle that predicts, given a domain, a problem and a metric, which planner will obtain the best performance and how long it will take. Given that it is not possible to have this oracle, in this thesis we propose the use of predictive models, automatically generated with Machine Learning (ML) and Data Mining (DM) techniques. ML is an area of AI that has evolved from the study of pattern recognition and computational learning theory (Russell and Norvig, 1995, Witten and Frank, 2005). It explores the study and construction of algorithms that can learn and make predictions on data. Notwithstanding, learning data should represent the planning task in our case. Previous works have already tried to represent planner performance, but they have only partially succeeded (Howe et al., 1999). Several of them are based on obvious features that could be extracted directly from the problem definition as the number of goals, number of objects, etc. Furthermore, other feature analyses have been carried out (Roberts and Howe, 2009, Roberts et al., 2008) but they still do not characterize the planning task properly. We hypothesize that characterization could be better and we propose new planning features, including features extracted from techniques that planners use to solve problems. These features could generate examples of data to use in a learning process, because the learning is always done through the observation of data, examples, direct experience, or instructions. In general, ML is about knowing how to do better in the future based on what has been experienced in the past, and representing that knowledge in models or other depictions. In our case, these models summarize the results of all the candidate planners from the past: whether they were able to solve planning problems or not, as well as the time required to generate a good solution (Cenamor et al., 2012, 2013). Given this knowledge on the past, the inductive hypothesis also gives us an idea as to how they will behave in future planning domains and with different problems. In addition, the order in which the planners are executed that can be given by the expected accuracy of these predictions. Therefore, we are able to configure a portfolio for each planning problem using these predictive models, like in previous works on the use of portfolios in search (Gomes and Selman, 2001). This is a renewed idea in automated planning since recent works have focused in static (Helmert, 2006) or domain-specific portfolios (Gerevini et al., 2009, 2014), in which the configuration of the portfolio is fixed for all the domains or chosen for each one respectively.

In this thesis we assert that ML can efficiently assist AP to build an Instance-Based Configured Portfolio (IBaCoP). IBaCoP is a family of planning portfolios that were built for competing in IPC-2014. One of these versions was the winner of the Sequential Satisficing Track, and is one of the main contributions of this thesis.

1.2 Objectives

This thesis analyses the state-of-the-art in planning portfolios and studies the mechanisms required to build, as the ultimate goal, a general system able to configure a planning portfolio using predictive models. To do so, this thesis focuses on several sub-objectives.

The first is the study of the possible planners: to create a portfolio, it is necessary to evaluate the previous planners to select the best candidates. This research takes into account the type of technique, its history and performance. However, given the large number of state-of-the-art planners, not all of them can be included in a portfolio. For this reason, this initial set of planners should be reduced. We analyze several planner filtering techniques in the state of the art, and we focus this study on coverage, time and quality score. Nevertheless, these mechanisms are not able enough to obtain a good selection as they may reduce the diversity of the set of planners. Instead, we propose a multi-criteria planner selection based on the Pareto dominance between time and quality to solve planning tasks.

The next objective is to develop a set of features able to characterize a planning task. The previous features in AP only focus on the elements that appear in the problem or domain definition, such as objects, types and goals. These kinds of features are not sufficient to characterize the planning task because problems with the same number of objects and goals could be extremely different, from a planner performance perspective. For example, a problem with only three blocks might be very easy, but as the Sussman Anomaly (Sussman, 1975) postulates, a particular instance of these kinds of problems could be hard for many planners because they typically work on goals separately. We argue that these kinds of features could be helpful but insufficient, and there is the need for other features that take into account the interactions and relationships within the search process. To improve the characterization, we propose additional features based on alternative representations of the problem, such as SAS$^+$, and another information extracted from landmarks, heuristics and the relaxed plan of the initial state. These features should be evaluated to understand what they are encoding, and how they are useful in selecting different candidate planners in a portfolio. This evaluation should include a comparative analysis of the performance between all features and their subsets. In addition, a study into what are the features that have different values in problems with similar structure is required. These problems have the same number of objects and goals when they are created but they may have other properties in the initial state that give rise to differences in the search for the plan.

The proposed set of features should be useful in creating more easily reusable knowledge with different predictive models. These predictive models are created to prognosticate the potential planners to carry out every planning task. In addition, these predictive models could be used to configure a portfolio. For this purpose, it is necessary to determine the best predictive model for the portfolio. To achieve this objective we propose to evaluate the predictive models following standard metrics from the field of machine learning.

The next sub-objective is to create several configuration strategies to build a portfolio. These strategies consist of selecting more than one candidate planner and assigning the running time. These strategies may improve the results of the predictive models when they are unable to discern the correct planner accurately or when all the planners are expected to behave similarly.

Another sub-objective is to verify the applicability of our proposal in different planning sub-fields. Although most of the research carry out in classical deterministic planning, we propose to characterize planning tasks also within a temporal planning setting. In this configuration, as in the previous one, the system would use the learned knowledge to configure a temporal planning portfolio, including the planner filtering, the predictive models and the configurable strategies. Portfolios for temporal planning were an unexplored idea at the beginning of this thesis, and there were none in the state of the art. In addition, the temporal planners are clearly separated into two different types, and, consequently, the selection of the candidate planners should be influenced by this planner division. This is a perfect situation in which to create portfolios, since the portfolio should include a combination of these types of planners. Following the hypothesis of this thesis, our study includes a characterization of the temporal task, including temporal specific features and, finally, strategies to build a temporal portfolio per instance.

Stated briefly, in this thesis we develop and update the idea of dynamic configurable portfolios that can be used for domain-independent or domain-dependent contexts.

1.3 Thesis Outline

This dissertation starts with its motivation, and a general overview on the reasons for creating planning portfolios based on knowledge acquired in previous planning process executions. In this first Chapter 1 (Introduction), we also explain the objectives of this thesis and we finish with the outline of this document.

Chapter 2, State of the Art, begins by describing Classical Planning, including the main formulations and representations of the planning task. This chapter follows with the description of the most important planning paradigms and the relevant planners for this thesis. A review of the portfolio construction techniques in AP, previous research based on the characterization of the planning task and the International Planning Competition (IPC) are included.

Chapter 3, The Planning Filtering Methods: A Multi-Criteria Approach, carries out an study of current selection techniques to maintain the most useful planners. This chapter includes a new proposal to filter the planners used in the planning portfolio and we compare all the approaches to understand their contributions.

Chapter 4, The Planning Problem Characterization and Empirical Performance Modelling, carries out a study on off-line learning to create predictive models. This chapter includes the description of current and new features developed to characterize the planning tasks. An evaluation of the learning of empirical performance models of the planners selected in the previous chapter is also reported.

Chapter 5, The Configuration Strategies to Create Planning Portfolios, provides an overview of the current strategies to configure planning portfolios, including some new proposals based on the use of empirical performance models. We present a study on the influence of the number of selected base planners in the performance of the portfolio and finally this chapter includes a portfolio construction algorithm that merges several proposed techniques.

Chapter 6, Experimental Evaluation of Instance-Based Configured Portfolios, performs the experiments to evaluate the deployment of the results in previous chapters to create planning portfolios. In this sense, we carry out a profound analysis of the results, showing which planners are the most selected and why they have been included. We also report several insights into the different versions of dynamic portfolios developed in this thesis, which obtained amazing results in different tracks of the IPC 2014 competition.

Chapter 7, The Performance Modelling of Planner in Homogeneous Problem Sets, develops a study into the relevance of the proposed features in homogeneous problems. This study confirms the hypothesis that a dynamic portfolio selects different sets of planners for different planning tasks and it gives us several insights into the relevance of the features in homogeneous problem sets.

Chapter 8, The Temporal Portfolio, explains the temporal approximation of dynamic portfolios based on empirical performance models. We create a temporal portfolio with a new set of temporal features and candidate planners, and we report the results when compared with current state-of-the-art planners.

Chapter 9 summarizes the results of this thesis, essentially presenting its most important contributions, gathering the main results achieved and proposing several lines of research that could continue this work.

In addition, this thesis includes four appendices: the planner components, a brief description of different methods of empirical evaluation in ML, training results for each planner component and the results of IPC-2014.

2

State of the Art

This chapter is a review of the background related to the main topics addressed in this thesis. First, we explain Classical and Temporal Planning. After that, we present the modelling languages in Automated Planning (AP) and an introduction to the International Planning Competition. Then, a set of AP concepts and techniques that are used in the following chapters is presented, since it is required to understand the features used to create empirical models, and how these features are computed. Specifically, heuristic planning, landmarks and SAT representations are described. This chapter finishes with the state of the art of planning portfolios and a report about current approaches for the characterization of planning tasks.

2.1 Classical Planning

Classical Planning (Ghallab et al., 2004) is one of the first approximations of AP. Summarizing, it deals with deterministic, static, finite, and fully observable state-transition tasks with restricted goals and implicit time. However there are other planning paradigms that do not assume some of them (e.g. uncertainty, temporal or probabilistic planning). Classical planning usually follows the STRIPS representation, which is described bellow.

Definition 1 (STRIPS planning task)**.** A STRIPS planning task is a state-transition system (the conceptual model in AP) described as a 5-tuple $\Pi = \langle \mathcal{S}, \mathcal{O}, c, s_0, s_\star \rangle$ formed by a finite set of facts, $\mathcal{S}$, a set of actions $\mathcal{O}$, a cost function c, the initial state s_0 of the task, where $s_0 \subseteq \mathcal{S}$, and a set of goals $s_\star$, where $s_\star \subseteq \mathcal{S}$. Each action $o \in \mathcal{O}$ is a triple (pre_o, eff_o, del_o) of subsets of $\mathcal{S}$ referred to the action's precondition, add and delete lists respectively, satisficing that $add_o \cap del_o = \emptyset$.

Π has a uniform cost if, for all $o \in \mathcal{O}$, $c(o) = 1$. The solution of a planning problem Π consists of generating a sequence of instantiated actions $(o_1, o_2, \dots, o_n)$, usually called a plan, where $o_i \in \mathcal{O}$. The application of a solution plan generates a sequence of states $(s_0, s_1, s_2, \dots, s_n)$ such that $s_i \subseteq \mathcal{S}$, $s_\star \subseteq s_n$, where s_i results from executing the action o_i in the state s_{i-1}, $\forall i = 1..n$. Finally, the cost of a plan is defined by $\sum_{i=1}^{n} c(o_i)$.

The representation of the STRIPS planning task is usually done through a :

- Domain model: it is the general description of how the world changes, normally defined as the set of allowed actions or rules that transform world states into other states.
- Problem: it is a particular situation of the domain, normally made up of a set of objects, an initial state, a set of goals that need to be achieved and a metric to be optimized.

Figure 2.1 details an example of a planning task in the transport domain. The transport domain consists of moving several packages from the initial state to the goals. Each vehicle (truck or airplane) can transport some packages depending on its capacity and moving has a cost depending on the length of the road. In this example, there are three vehicles, one package and four locations in two different cities. Each truck can drive only in the same city and airplanes fly between cities. The goal of this problem is to have the package in a particular location. More specifically, in this problem, the initial state (s_0) contains the static information about: the connections between the locations in the same city (loc-01 with loc-00 and back); and the connections between different cities (loc-01 with loc-10 have one). Furthermore, the initial state also contains dynamic information such as: where the package is located (*at p loc*-00); where the trucks in the two cities are (at loc-01 and loc-11); and where the plane is (*at a loc*-01). The domain definition has the actions of move, load and unload from the different vehicles.

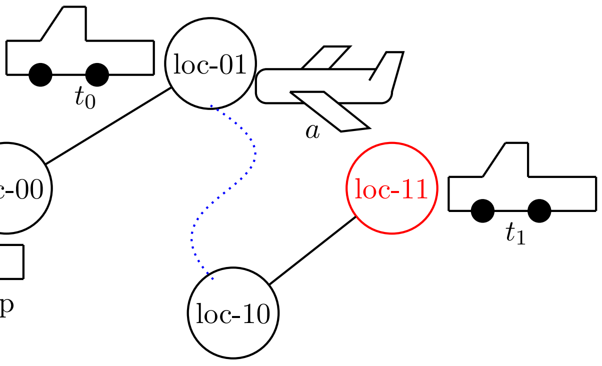


Figure 2.1: Initial state in a transport problem with a package, two cities, two trucks, and a plane. The location in red is the final state of the package. The road for each truck is the continuous line and the available movement for the airplane is the blue dotted line.

The plan for this problem is an ordered sequence that includes the actions: to move truck t_0 from loc-01 to loc-00, to load the package p, to move the truck back to loc-01, to unload the package p, to load the package p onto the airplane a, to fly from loc-01 to loc-10, to unload the package p, to move the trunk $t1$ to loc-10, to load the package p, to move the truck back to loc-11 and to unload the package p. The location of the vehicles in the goal state is not important, only the final location of the package.

2.2 Temporal Planning

Classical planning imposes several assumptions, such as the implicit time of the actions (Gerevini et al., 2006). In many real-world planning domains, the carrying out of certain actions can only occur during some predefined time windows where one or more necessary conditions is imposed. For instance, a truck can be refueled at a gas station only when the gas station is open. Temporal planning arises with the reason for handling situations where actions have variable duration, might overlap in time or an explicit representation of time is required.

Definition 2 (Temporal Planning task (Fox and Long, 2003))**.** A temporal planning task is a tuple $P = \langle \mathcal{S}, \mathcal{A}, c, s_0, s_\star \rangle$, where $\mathcal{S}, c, s_0, s_\star$ are defined as in classical planning in Definition 4.3.5. However, each $a \in \mathcal{A}$ is a temporal action made up of:

- duration $(d(a))$,
- $pre_s(a)$, $pre_o(a)$, $pre_e(a)$: preconditions of a at the start, over all and at the end, respectively,
- $add_s(a)$, $add_e(a)$: add effects of a at the start and at the end,
- $del_s(a)$, $del_e(a)$: delete effects of a at the start and at the end.

A plan for P is not a sequence but rather a set of action-time pairs $P = (a_1, t_1), ..., (a_n, t_n)$, where $t_i, 1 \leq i \leq n$, is the scheduled start time of action a_i. Let $(a, t)\ in\ P$ be an action-time pair of the plan. Although a has a continuous duration, its effects only apply at the start and the end. We can therefore associate two discrete events $start_a$ and end_a to a, such that $start_a$ has associated time t and end_a has associated time $t + d(a)$. The plan P induces an event sequence $P = \langle e_1, ..., e_{2ni} \rangle$ that includes $start_a$ and end_a for each pair $(a, t) \in P$ and is ordered by the associated times of events. We only consider plans that associate single times to events, i.e. actions cannot start and/or end simultaneously.

The temporal actions present events $start_a$ and end_a as classical actions (Celorrio et al., 2015) to describe the semantics of a temporal plan:

1. $pre(start_a) = pre_s(a)$, $pre(end_a) = pre_e(a)$,
2. $add(start_a) = add_s(a)$, $add(end_a) = add_e(a)$,
3. $del(strat_a) = del_s(a)$, $del(end_a) = del_e(a)$.

There are two types of temporal planning (Cushing et al., 2007). The first, conservative temporal planning, in which concurrency is strictly optional. These problems have some additional temporal information (action durations) that can be used to obtain the scheduling of the sequences. The second is interleaved temporal planning, in which concurrency is required; the actions in the plan could be carried out in parallel and it is important to fulfill the requirements (preconditions, add and delete *at start*, *at end* or *over all*) of actions at specific time. Figures 2.2 details a graphical representation of the action $Move$ from one location $?l1$ to $?l2$. To start the action, it needs the truck

to be in location $?l1$, that both locations are connected, and that there is enough fuel.
At the eased
by the

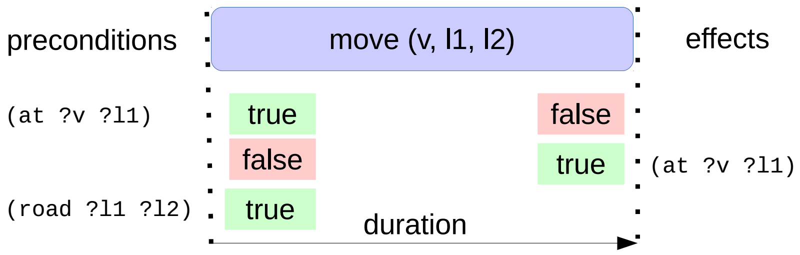


Figure 2.2: Graphic representation of a durative action temporal transport domain

As noted above, Figure 2.2 details the same action graphically. It represents when the facts are true and false in the course of the actions, and when a fluent changes the value (in this case decreases).

2.3 Modeling Languages in Automated Planning

A common specification language is useful to define the classical planning problem. The planning community has adopted Planning Domain Description Language (PDDL) (Ghallab et al., 1998) as the *de facto* standard (Rintanen, 2015). PDDL appeared in 1998 and this standard has promoted several comparisons between planning algorithms for classical planning. This fact led to the development of standard benchmark domain and problem sets. However PDDL is not the only way of formalizing a planning task. Several planning systems formulate the planning task with other representations on the basis of the techniques used. The SAS$^+$ representation (Bäckström and Jonsson, 1995) is a transformation of the propositional definition in PDDL that defines a set of finite multi-valued variables in contrast to PDDL which defines a set of Boolean propositions. The translation from PDDL to SAS$^+$ can be performed automatically (Helmert, 2009). The finite domain representation is presented in detail in subsection 2.3.2.PDDL is described bellow.

2.3.1 PDDL

PDDL was proposed as the language of the first International Planning Competition, whose objective was to compare state-of-the-art techniques and further encourage the development of more efficient planners. This standard has evolved with several versions, each of them adding new features.

An example of a problem and domain definition appears in Figure 2.3. The first part of the figure sets out the definition of the domain. This file represents the predicates and the actions that model this world. The second part is the example described in Figure 2.1 for this domain in PDDL.

We highlight two versions of PDDL below:

```
(define (domain transport)
  (:requirements :typing :action-costs)
  (:types
        location target locatable - object
        vehicle package - locatable
        capacity-number - object)
  (:predicates
     (road ?l1 ?l2 - location)
     (at ?x - locatable ?v - location)
     (in ?x - package ?v - vehicle)
     (capacity ?v - vehicle ?s1 - capacity-number)
     (capacity-predecessor ?s1 ?s2 - capacity-number))
    (:action drive
    (:action pick-up
    (:action drop
...

(define (problem transport-cities)
      (:domain transport)
        (:objects
          city0-loc0-0, city0-loc0-1, city1-loc1-0, city1-loc1-1  - location
          truck0, truck1, airplane0 - vehicle)
          package - package
        (:init
          (at truck0 city0-loc0-1)
          (at truck1 city1-loc1-1)
          (at airplane0 city0-loc0-1)
          (at package city0-loc0-0))
        (:goal
          (at package city1-loc1-1))
```

Figure 2.3: Simplified example of the transport problem and domain in which the connection between locations and actions are omitted. This problem is the same as appears at Figure 2.1

- PDDL 1.7 (Ghallab et al., 1998) supports the basic representation (STRIPS (Fikes and Nilsson, 1971)), using a LISP-like syntax. This version includes action preconditions and the goals can be expressed not just as conjunctions of propositions but also as disjunctions or quantified formulas. In addition, actions can have conditional effects and typed objects.

- PDDL 2.1 (Fox and Long, 2003) added numeric variables which could be modified by actions. A new type of action is included: durative actions (for temporal planning). Durative actions could have discrete and continuous effects. These actions allowed time on the planning task to be represented. Figure 2.4 details a durative action.

```
(:durative-action move
     :parameters
          (?v - vehicle ?l1 ?l2 - location)
     :duration
          (= ?duration (road-length ?l1 ?l2))
     :condition (and
          (at start (at ?v ?l1))
          (at start (road ?l1 ?l2))
          (at start (>= (fuel-left ?v) (fuel-demand ?l1 ?l2)))
     :effect (and
          (at start (not (at ?v ?l1)))
          (at end (at ?v ?l2))
          (at start (decrease (fuel-left ?v) (fuel-demand ?l1 ?l2)))))
```

Figure 2.4: Description of durative Action Temporal Transport Domain

Figure 2.4 details an action in which a vehicle can drive between two points. During the trip, it uses up fuel and the time depends on the length of the road. It is important to note that the precondition should be true when the action starts (*at start* precondition). Other actions can restrict some preconditions throughout the entire duration (*over all*). The effects in this action could be applied *at the start* and *at the end*.

2.3.2 SAS$^+$

The SAS$^+$ planning formalism (Helmert, 2004) is an alternative to propositional STRIPS (Fikes and Nilsson, 1971) that has been analyzed in depth by several researchers. Unlike STRIPS, it allows state variables to have non-binary (finite) domains. Planning tasks specified in propositional STRIPS can be easily translated into a SAS$^+$ planning task (Helmert, 2009). The definition of the planning tasks with multi-valued variables is shown in Definition 3, which is shortened to the FDR Planning task.

Definition 3 (FDR planning task)**.** A deterministic planning task in finite-domain representation or FDR planning task is a 5-tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, c, s_0, s_\star \rangle$, where:

- $\mathcal{V}$ is a finite set of state variables, each $v \in \mathcal{V}$ with a finite domain $\mathcal{D}$.

- $\mathcal{O}$ is a finite set of actions, each $o \in \mathcal{O}$ is a pair (pre_o, eff_o).
- c is the cost function.
- s_0 is a complete variable assignment called the initial state.
- $s_\star$ is a partial variable assignment called the goal.

This formalism permits the construction of two different types of graph; a Causal Graph (CG) and a Domain Transition Graph (DTG). The causal graph is a directed graph that represents the interaction among variables in the planning task. Each node is associated with a finite-domain variable and there is an edge from p to q if the value of variable q depends on the value of p. It was first used in the context of the generation of hierarchical abstractions (Bacchus and Yang, 1994, Knoblock, 1994). Later, it was defined for unary operator tasks (Brafman and Domshlak, 2003). We use the definition given by Helmert (Helmert, 2004) in the context of heuristic search planning:

Definition 4 (Causal Graph)**.** Given a FDR planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, c, s_0, s_\star \rangle$, a **causal graph** is a directed graph $(\mathcal{V}, A)$ containing an arc (u, v) if and only if $u \neq v$ and there exists an operator $(pre(o), \mathit{eff}_o, c(o)) \in \mathcal{O}$ such that $v \in \mathcal{V}_{\mathit{eff}_o}$ and $u \in \mathcal{V}_{\mathit{eff}_o} \cup \mathcal{V}_{pre(o)}$.

The causal graph of a planning task offers an overall view of the interaction among variables in which the high level variables in this graph are the variables that define a goal in the problem. The weight for the arcs is the number of DTG transitions that are included in each arc.

Figure 2.5 shows an example of the Causal Graph in a Transport domain referred to in Subsection 2.3.1. In this figure, there is only one high level variable and it appears in red. This high level variable is the package p_0 in the problem, and the other variables are the trucks and the airplane. For each arch between a vehicle and the package, the graph represents the possible actions in the domain; e.g, the loading and unloading of the package in the available positions (two in all cases).

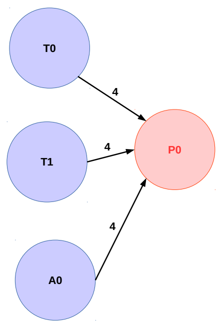


Figure 2.5: Causal Graph on Transport Domain. It represents two trucks, an airplane and a package.

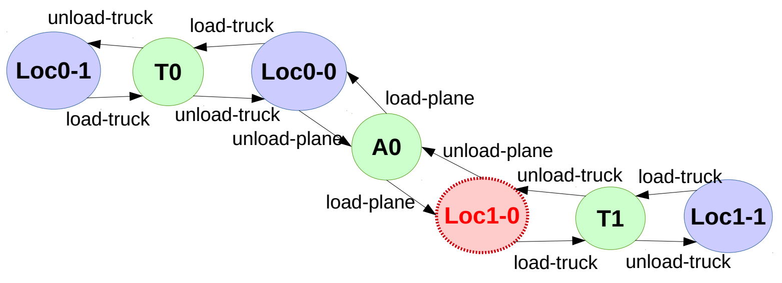
Furthermore, domain transition graphs (DTG) describe the transitions between values of a single variable of the problem, and their relationship with other variables (Jonsson and Bäckström, 1998).

Definition 5 (Domain Transition Graph)**.** A **domain transition graph** of a state variable $v \in \mathcal{V}$ of a FDR task $\Pi = \langle \mathcal{V}, \mathcal{O}, c, s_0, s_\star \rangle$ is the labeled directed graph $DTG(v, \pi)$ with vertices $\mathcal{D}_v$ and an arc (d, d') labeled with action $o \in \mathcal{O}$ if and only if $d \neq d'$ and there is an operator $(pre(o), eff(o), c(o))$ where $(pre(o) = d$ or $v \notin pre(o))$ and $eff(v) = d'$. For each arc (d, o, d') we say that there is a transition of v from d to d' enabled by o.

Figure 2.6 details the DTG for three variables in the previous example. It is important to note that the arc representations in this figure are a simplification of the instantiated actions that appear in the SAS$^+$ representation. For example, in the sub figure 2.6a the instantiated action is *drive-package0-city1* because there are other objects which are parameters of the action. As in the previous figure, the goals in this
ed. The weights in the nditions
words, the number of specific

Drive
Loc1-0
Loc1-1
Drive
Fly
Loc0-0
Loc1-0
Fly

(a) DT



(c) DTG package 0

Figure 2.6: Three DTG's on Transport domain. The graph (a) depicts the transitions of a single truck in city 1 (the DTG for the other truck is the same but changes the name of the nodes with the names of the locations in the city 0). The graph (b) is the transitions of an airplane. The graph (c) depicts the transitions of a package in the transport domain where the green nodes represent the vehicles.

Both modeling languages (PDDL and SAS$^+$) are used in next chapter to generate features and characterize the planning task. More specifically, the structure of the graphs is one of the most relevant knowledge sources in this dissertation.

2.4 The International Planning Competition

The International Planning Competition was held for the first time in 1998 to obtain a general framework to compare the current planning systems and to assess the state of the art. The competitions have several advantages for the planning community: planning benchmarks are changing; the created planning systems are improving remarkably every year; planners are capable of solving large and complex problems, using rich expressive domain models and improving the quality of solutions. In addition, the IPC determines an empirical methodology to compare the planners and implant several metrics. These metrics include coverage, quality and time, which are described below:

- **Coverage**: The coverage is the total number of problems solved from the benchmark within the time and memory bounds.
- **Time score**: While coverage is only taken into account if a planner solves a problem or not, time score metrics reward planners for solving the problems as soon as possible. Thus, the planner which solves each problem in less time receives one point, while other planners solving the same problem in more time receive a score of between 0 or 1. The specific formula is shown in equation 2.1. T^* is the minimum time required by any planner to solve the same task and T is the time this particular planner took to solve the task. If the planner does not solve the problem, then $T = 0$.

$$T = \frac{1}{1 + \log(\frac{T}{T^*})} \tag{2.1}$$

[1]

- **Quality score**: A planner finds a plan with a quality where $Q < Q^*$ (assume the best quality is Q^*, and the obtained quality is Q). This planner would aggregate the quality ratio Q^*/Q to its total score (lower qualities are considered best). In cases in which it is not feasible to obtain Q^*, it will be considered the best plan quality found by any planner and, like the previous metric, if a planner does not solve the problem, $Q = 0$.

The last two metrics are scores that depend on the included planners; it is meant that the T^* and Q^* are calculated with the execution of the planners because an optimal solution could be computationally expensive. In addition, the competitions have several rules that contribute to standardizing the process of evaluating automated planners. The benchmarks are described in the Planning Domain Definition Language (PDDL) (Ghallab et al., 1998), which is used to provide domain and problem descriptions. Every IPC developed a software that automates the experimentation (López et al., 2013). In this thesis, the experiments are carried out within the framework of the IPC-2011. The software automatically tracks the execution of the planners, measuring the time and memory use. In addition, after the execution of each planners, the results are validated by an automatic validation tool (VAL) (Howey et al., 2004).

Every IPC was made up of several tracks; for example, IPC-2011 had three (deterministic, learning and uncertainty). However, each track, at the same time, is split

[1]This formula is assumed for our evaluation. The first time that this formula is used in learning track. The time score is given by T^*/T.

into categories. The deterministic track is divided into: temporal and sequential; and sequential has satisficing, optimal and multi-core tracks. In addition, in the last IPC, the agile track was included in this section. The goal in every track is different and, consequently, the requirements and metrics might be different. We highlight several tracks below:

- The **Satisficing track** covers classical STRIPS planning (non-durative actions) with actions having associated non-negative costs (not necessarily uniform), negative preconditions and conditional effects. The goal of this track is to find low-cost plans, and the planners can generate more than one solution, where the cost is defined as the sum of the costs of each plan's actions. The metric for this track was the quality score and the memory limit was 4 GB in the last IPC.

- The **Optimal track** seeks to find optimal plans in terms of total cost, where the total cost of each plan is defined as the sum of the costs of its actions. Planners compete on the number of problems they manage to solve (coverage).

- The **Multi-core track** takes into account that the code of the planner will run in different cores simultaneously and/or with different threads in each core. No GPU computing is available. Moreover, it is not the aim of this track to distribute work among different nodes of large clusters. Only one computer with a number of cores is devoted to each planner. And the memory of all cores is the same as the satisficing track. The metric for this track is the quality score.

- The **Agile track** objective is to find a satisfacing solution as soon as possible, given the very short amount of CPU time available. The metric appears on the formula for the time score described in Equation 2.1.

- The **Learning track** is a track made up of two stages. In the first stage, the system extracts Domain-specific Control Knowledge (DCK) for each domain. The source code should be prepared previously to support this knowledge and the system is evaluated in each domain of the competition with and without the learned DCK on the same problem set. The no-knowledge evaluation helps to provide insights into the impact that learning had for each participant. The metric for this track is the quality score.

- The **Temporal track** consists of finding short plans in terms of their makespan. The planners in this track have to support durative actions and numeric state variables. The metric for this track is a quality score.

2.5 Planning Paradigms

The AP community has contributed many algorithms that are implemented for domain-independent planning. In this section, we explain heuristic planning as the most extended technique, landmarks and SAT.

2.5.1 Heuristic Planning

There is extensive related work on planning with heuristic search in which the search algorithms try to find a path in a graph between the initial state and the final state (a state that satisfies the goals). An example of a planning graph appears in Figure 2.7. This graph has a group of eight states and the only path to achieve the goal state is $\langle Init, S_2, S_4, Goal\rangle$.

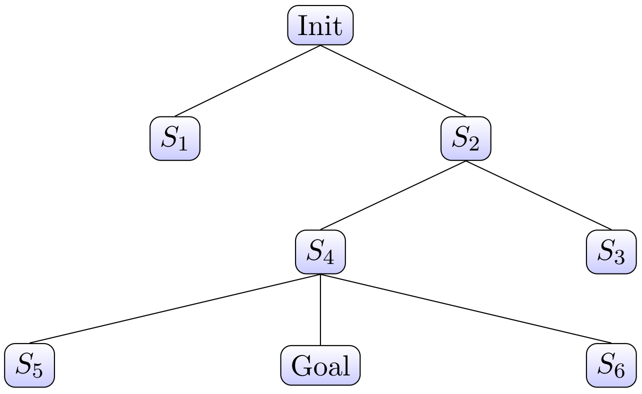


Figure 2.7: Example of a graph plan

In order to improve the search algorithms, heuristics introduce a guide to go through a planning graph, skipping useless nodes in the graph, and making the solution finding easier for the planning problem. Typically, a heuristic function estimates the remaining cost from the current state to the goals. It is usual to denote the heuristic evaluation function by $h(s)$.

Definition 6 (Heuristic function)**.** A heuristic function $h(s)$ estimates the cost of a path from the actual state s_i to the goal; search methods prefer to expand states s_{i+n} with small $h(s_{i+n})$ values.

The heuristic search-based planners can be categorized by following three different criteria: the direction of the search (progression from the initial state or regression from the goal state); the search algorithm used (the most extended are best-first and hill climbing search); and the heuristic function used for the search.

Heuristic Search Planner (HSP) (Bonet and Geffner, 2001) is one of the pioneer planners that uses heuristic functions. This planner applies a hill-climbing search with heuristic h_{add} from the initial state to the goal (progression search). This heuristic is based on a delete-relaxation of the problem (a version of the original planning task in which all the *delete* effects of all the actions $a \in A$ are ignored). h^{add} is computed by iteratively adding the costs of achieving the preconditions of the actions that get the goals.

Fast Forward (FF) (Hoffmann and Nebel, 2001) uses progression search in the state space with a heuristic that estimates goal distances by ignoring delete lists (h^{ff}). This planner introduced several techniques that increased its efficiency considerably with respect to the previous planners. For instance, FF has a more sophisticated method for heuristic evaluation than HSP; it uses a kind of local search strategy, employing systematic search for escaping plateaus and local minima and it identifies the successors that are most helpful in achieving the goal: helpful actions. The set of helpful actions are the set of useful successors generated by the relaxed plan, where the relaxed plan is an instance just like the plan Π, but in which the operators do not have a delete list.

Fast Downward (FD) represents a generation of heuristic planners that use the progression search (Helmert, 2006). FD is a planning framework that uses a multi-valued representation (SAS^+) of the problem (Helmert, 2009) and implements a broad range of techniques. The most successful ones include landmarks, new heuristics (Causal Graph, Landmark, etc.) and the use of multiple open queues, amongst others. There are several planners that are implemented on top of FD. For example, planners such as LAMA-2008, LAMA-2011 and portfolios like FDSS. In this section, we will explain several techniques that are based on this system (landmarks, SAS^+ formulation) and we use them in this thesis.

FD carries out a planning task in three phases as detailed in Figure 2.8. The first step is the *translation* process. This step transforms PDDL into a multi-valued representation. The second step, *preprocess*, generates structures in the system, such as Causal Graphs and Domain Transition Graphs. The last step, *search*, implements three different search algorithms with different heuristics.

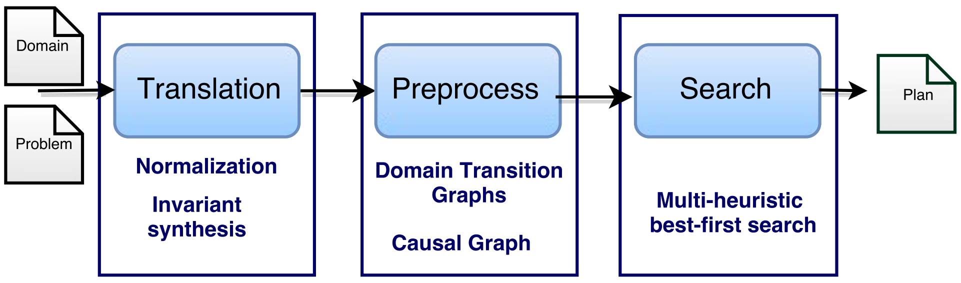


Figure 2.8: The three phases of FD's execution.

2.5.2 Landmarks

In its initial definition, landmarks are a set of propositional formulas that have to become true at some point in every plan (Porteous et al., 2001). Later, that definition was extended to include both action landmarks (Richter and Westphal, 2010) and conjunctive sets of propositions (Keyder et al., 2010).

Definition 7 (Fact Landmark)**.** Given a FDR planning task $\Pi = \langle \mathcal{V}, \mathcal{O}, c, s_0, s_\star \rangle$ and $s \in \Pi$. A fact p is a fact landmark for s if $p \notin s$, and for every plan $(o_1, o_2, \ldots, o_n)$ for s, there exists an state t so that $p \in apply(s, \langle o_1, o_2, \ldots, o_t \rangle)$.

Definition 2.5.2 explains that a fact landmark is a variable value that must become true at some point during every plan, but not in the current state s. This landmark should appear on the eff_o of the actions of the plan $(o_1, o_2, \ldots, o_n)$.

Landmarks were also formalized in a framework that relates them to abstractions and critical paths (Helmert and Domshlak, 2009), giving them a stronger theoretical background. Several planners use landmarks to direct the search towards those states where many landmarks have already been reached. The landmarks are the intuitive solutions that the humans might use. For example, consider the example in Figure 2.1 where the goal is to deliver packages to various locations using different vehicles. Cities consist of sets of locations, where trucks may transport packages within the city, whereas airplanes have to be used between cities. For a human, the first step is to load the package onto the truck to move it to the airport, because the package must be transported between the two cities, from city 0 to city 1.

The landmark precisely captures these intermediate conditions which can then be used. Figure 2.9 shows several landmarks, *L1 = "package at loc-01"* and *L2 = "package at in loc-11"* are examples of landmarks in the task shown in Figure 2.1. Furthermore, these landmarks give extra knowledge because *L1* must be true before *L2*. The entire this process could be carried out in a pre-processing step (Hoffmann et al., 2004).

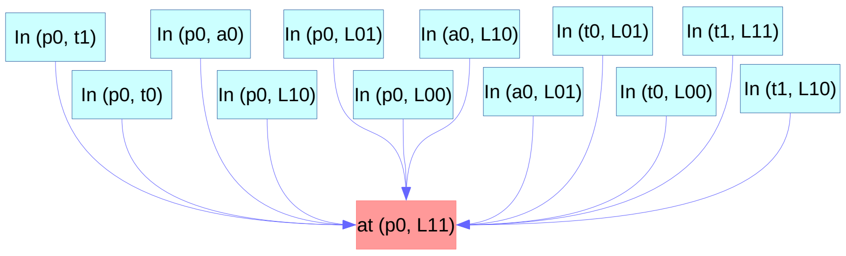


Figure 2.9: Landmark graph produced by Zhu & Givan method from the previous problem in Figure 2.3 at transport domain.

More specifically, Figure 2.9 shows the generation method by Zhu & Givan (Zhu and Givan, 2003), in which the landmark graph represents a conjunction of propositions (the preconditions of the actions). In this graph, the landmarks that appear in the relaxed problem after a simple verification process is carried out. This representation is an easy representation of the landmarks. This graph connects all facts with the fact that achieves the goal *at (p0, L10)*.

This representation is not the only one: Richter, Helmert and Westphal (Richter et al., 2008) introduce another. Their algorithm recognizes landmarks and landmark orderings for a multi-valued state variable representation of planning tasks.

A more recent approach (Keyder et al., 2010) creates landmarks for the delete relaxation of the planning problem. This method does not guarantee to find all the possible sets of landmarks. However, this method guarantees that all the landmarks in accordance with the causality criterion will be found. This method is similar to the previous one but adapted to the SAS$^+$ formulation to create more accurate landmarks with better orderings. Figure 2.10 details an example of a landmark graph. There

is real difference b one shows more
dependencies betwe f facts.

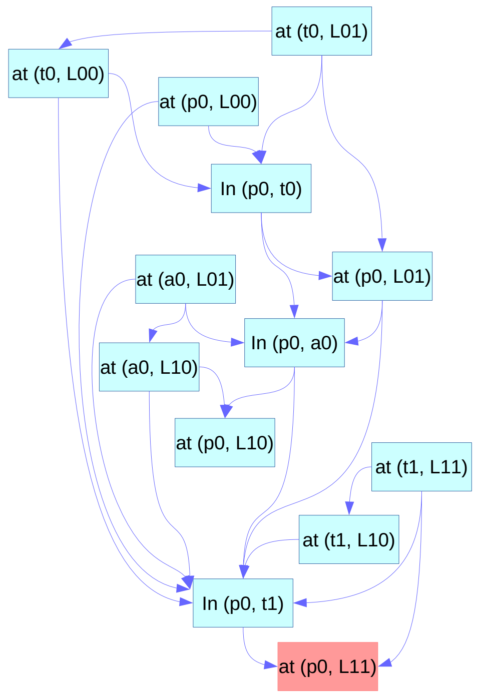


Figure 2.10: Landmark graph extracted by the formulation of Keyder, Richter & Helmert with $m = 1$ from the previous problem in Figure 2.3 at transport domain.

2.5.3 SAT

Propositional satisfiability (SAT) is the problem of determining whether there is a substitution of variables for specific values in a propositional formula that make the formula true. SAT is one of the first problems proven to be NP-complete and one of the most important areas in AI. Planning as satisfiability is an approach to domain-independent planning and is one of the earliest applications of SAT to solve generic search problems. In this scope, the planning task becomes an to satisfy a propositional formula which can be easily constructed from the problem description (Biere et al., 2009). The idea of using SAT for planning is to encode the planning problem as a bounded plan existence problem, i.e., whether a plan of a given length exists, as a formula in SAT. The efficiency of the SAT-based approach to AI planning is determined by an efficient representation of the problem.

The final goal of these techniques is to determine whether there is an interpretation that satisfies a given Boolean formula. This is a domain-independent planning approach proposed by Henry Kautz and Bart Selman in SATPlan (Kautz et al., 2006). This planner compiles a planning task iteratively into several SAT tasks using the concept of parallel steps. The planner maps the problem into a SAT instance and uses a state-of-the-art SAT solver to find a solution or prove that there has been no solution up to the current horizon. This planner has another version, SatPlan-2005, which includes a mutex propagation. This process is carried out on the plan graph but only a subset of the inferred mutexes are encoded as binary clauses. An encoding with Boolean variables for both actions and fluents is used, rather than on which only uses actions.

The current state-of-the-art SAT-based planner is Madagascar (Rintanen, 2014). This planner includes efficient encoding, parallelized search strategies (Rintanen et al., 2006), and SAT heuristics specialized for planning. There are three versions of this planner, Madagaspar, Madagascar-p and Madagascar-pC. Madagascar-p is similar to Madagascar but it uses backward chaining and Madagascar-pC is similar as Madagascar-p but it replaces the horizons that uses the others for smaller ones.

2.6 Portfolios

In this Section, first, we set out some relevant definitions on planning portfolios. Next, we describe the different portfolio configuration strategies that can be found in the literature, and we characterize them by specifying some important properties. Finally, we describe the state-of-the-art techniques in the planning portfolios.

2.6.1 Portfolio Definition

The automated planning community has created a very large number of different planning algorithms (solvers). No single solver dominates all of the others on all planning tasks, although there is an overall winner in each IPC. For this reason, combining all of them could be a good idea to achieve a better overall performance than any base one. However, there are many ways of defining a planning portfolio. A simple way is to assign run time slots to each base planner, so it must only be executed in these time slots. Another way is by assigning execution times to a sequence of planners, as we do in Definition 8.

Definition 8 (Planning Portfolio)**.** Given a set of base planners, $\{pl_1, \ldots, pl_n\}$, and a maximum execution time, T, a planning portfolio can be considered as a sequence of m pairs $< pl_1, t_1 >, \ldots, < pl_m, t_m >$. where $pl_i \in \{pl_1, \ldots, pl_n\}$ and $\sum_{j=1}^{m} t_j \leq T$.

The goal of a portfolio is to offer an implementation of the Virtual Best Solver (VBS), described in Definition 9. The VBS selects the best solver from a given portfolio of alternatives on a per-instance basis (Cameron et al., 2016). This approximation usually achieves much a better performance than the Single Best Solver (SBS), SBS being the best planner for a set of tasks. This VBS provides a useful theoretical upper bound on the currently achievable performance. However, this bound is usually not tight: the VBS is not an actual planner (to compute without an oracle, it needs to run each planner sequentially) and the VBS is calculated by a set of planners. As a consequence, VBS is biased by the planners used, it is not generalizable for new instances.

Definition 9 (Virtual Best Solver (VBS))**.** Given a set of base planners, $pl_1, \ldots, pl_n$, a set of problems to solve, $\{p_1, \ldots, p_k\}$, and a metric m, the Virtual Best Solver is the best possible combination of planners to solve the set of planning tasks: $VBS = \arg_{i \in \{1,\ldots,n\}} \min m(pl_i, p_j); \forall j \in \{1 \ldots k\}$.

VBS is, therefore, an oracle which always selects the best planner for each planning task and it always achieves the upper bound in terms of the metric used.

2.6.2 Portfolio Construction Process

There are several steps in configuring a portfolio of planners (Vallati, 2012) as described in Figure 2.11.

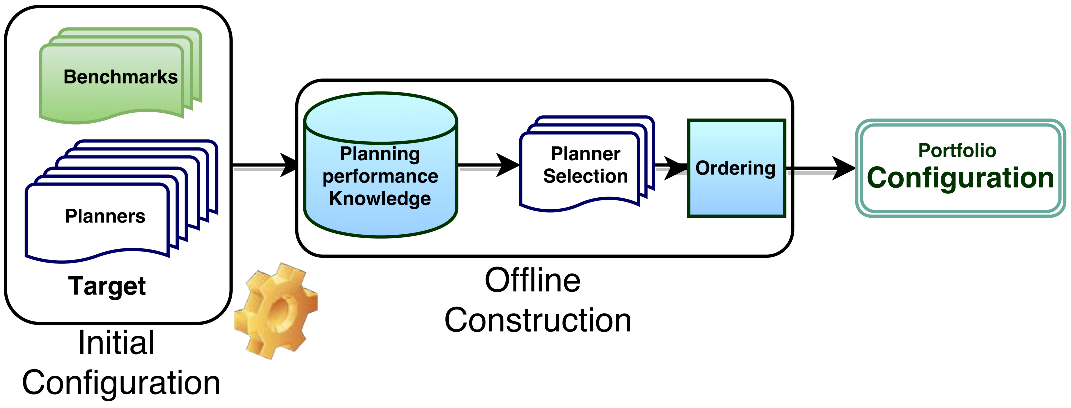


Figure 2.11: Steps required for configuring a planning portfolio.

The first step is to select the **initial configuration** from which the portfolio will be constructed. This information consists of the set of base planners, the planning benchmarks, and portfolio configuration data. An accurate selection of base planners is a main issue when constructing portfolios, since the portfolio will only be able to solve problems that the candidate planners can solve but nobody else. However, the number of candidate planners should be bounded because the total execution time is distributed among all the planners, so more planners mean less average time per planner. And limitations in time assignment may increase completeness problems.

The task benchmark influences the target of the portfolio. For example, if the training set only includes problems in the same domain, the portfolio achieved is domain-dependent. However, if the benchmark suite is too large, the process of analysing the benchmark may require a great computational effort, which may also increase with the number of planners. In this scope, there are not an standard criteria on how the domains/problems are to be selected; the most widespread is to use IPC benchmarks to make up the training set.

Other decisions are related to the type of portfolio we want to create, such as *configuration target*, *optimization criteria*, or *order*. The configuration target defines whether the portfolio is domain-independent (static), domain-specific or instance-specific. The optimization criteria may include several factors,such as the time available, whether this time is fixed or whether we want maximize coverage or minimize execution time. Execution order of the base planners is also an important issue, and also whether the portfolio can be executed sequentially or in parallel.

The second step is **offline construction**. All selected planners are run in the training benchmark. This process generates a database with "Planning Performance Knowledge". In previous works, this knowledge was mainly used only to establish the running time per base planner and the relative order in the portfolio. However, in this thesis, this information is also used to provide an initial planner selection.

Lastly, the final selection of planners, their execution time and relative order defines a specific **portfolio configuration**. Figure 2.12 shows a portfolio in which there are three components selected to run for a fixed amount of time. If the order of the sequence is assumed, the portfolio will present an arbitrary order of execution (e.g., Pl6, Pl11, Pl3). Note , there are no constrains on a planner appearing different points of the sequence, although the portfolios that we build in this thesis satisfy this constraint ($\forall pl_i, pl_j \in < pl_1, t_1 >, \ldots, < pl_m, t_m >, pl_i \neq pl_j$, therefore $m \leqslant n$, where m is the number of selected planners and n is the initial one). Other assumptions of our approximation are that: i) there is no transfer information between each candidate planner at the time of execution; ii) each candidate is a black box in the portfolio, as they appear in Figure 2.12.

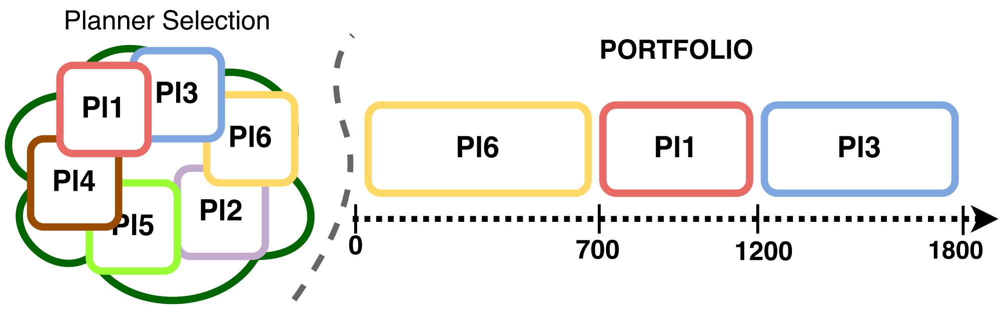


Figure 2.12: Example of sequential portfolio

2.6.3 Planning Portfolios

In this section, we review the main portfolios in the literature. To describe them, we use a set of characteristics summarized in Table 2.1. This characterization includes whether the portfolio has a step to filter the planners or its selection is quite arbitrary (F); how the time is assigned to each base planner (T), i.e. whether it is a uniform assignment, based on quality metric rankings, etc; the order in which each component is executed (O), performance prediction, expected required time, etc; the type of portfolio configuration (C): static, dynamic per domain or dynamic per problem; whether the portfolio has components that could transfer information between themselves (Pl);

whether the planner is domain independent or not (K); and last, in which IPC track the planner has participated or is designed for.

Table 2.1: Different features of planning portfolios.

Characteristic	Possible values
Number of Planner candidate (N.)	Natural number
Planner Filtering (F)	Yes / No
Time Assignment(T)	Equal Time (Uniform), based on score, etc.
Order (O)	Performance, less time, etc
Configuration (C)	Static / Dynamic / Per problem
Independent planners (Pl)	Yes / No
Knowledge (K)	Other information
Track	Sequential Satisficing, Optimal, Agile, Multi core, Temporal, Learning

Howe et al. describes one of the first per instance planning portfolios (Howe et al., 1999) . They implement a system called *BUS* that runs 6 planners (all planners available until that moment) and whose goal is to find a solution in the shortest period of time. To achieve this, 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 in accordance with the estimation provided by a linear regression model of their success and run-time so, they use predictive models for the behavior of the planners to decide their order of execution. They use only 5 features extracted from the PDDL description: from the domain, they count the number of actions and the number of predicates; from 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 an instance-specific configuration with independent planners.

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 for selecting and combining heuristics and search algorithms. In training, they evaluate the possible configurations with a time limit, and select the set of configurations that maximizes the IPC score. For the portfolio presented in the IPC-2011 Sequential Satisficing Track, they sort the configurations by decreasing the order of IPC score, 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. However, it is not the only ordering strategy, FDSS for optimal track selects the components that maximizes the coverage and sorted by decreasing the order of failures in memory use. These portfolios present a sequential domain independent configuration. They share the best solution among the candidate planners. The candidate planners, their order and their time are designed by empirical evaluation.

Fast Downward Cedalion (Seipp et al., 2015) is a portfolio configured with an automatic parametric algorithm. Given a parametric planner and a set of training instances, it selects the pair (planner, 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 SMAC (Hutter et al., 2011), which is a model-based algorithm configurator on the training instances. *Cedalion* has the same configuration for all the problems but a different configuration per version (There are 4 versions of Cedalion: Satisficing, optimal, agile and learning). The diversity of the candidate planners is limited by the heuristics and search methods included in FD. 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 et al., 2009) to find fast configurations of the FD planning system for 21 planning domains separately. At run time, all configurations found are run sequentially for the same amount of time for 85 seconds at most.

PbP (Gerevini et al., 2009) is a domain-specific portfolio with independent planners. 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 (Gerevini et al., 2014)). The automatic portfolio configuration in *PbP* aims to build different types of planning systems: a domain-optimized portfolio planner for each given domain in *PbP*. It uses several planners that focus on macro-actions. This portfolio was the winner of the learning tracks of IPC-2008 and IPC-2011 and the additional knowledge of the domain is extracted in a previous offline phase. Furthermore, PbP has two different variants one focuses on time, and the other focuses on quality. This portfolio is built focud on learning track, but it could be used in a domain-independent way (not including the offline learning knowledge per domain).

MiPlan (Núñez et al., 2015a) is a sequential portfolio using Mixed-Integer Programming (MIP), 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. For this approximation, the planner does not consider the other portfolios, only their components. This portfolio has four versions: optimal, satisficing, multi-core and learning. The resulting portfolio is a linear combination of planners defined as a sorted set of pairs $planner, time$ without sharing information among them. Their MIP model considers an objective function that maximizes a weighted sum of different parameters including overall running time and quality per track.

DPMPLAN (Núñez et al., 2014) is a modified portfolio based on MIPlan. In this case, the MIP task tries to maximize the objective function for a set of times (in seconds). The times considered are 1, 5, 10, 25, 50, 100, 200, 450, 900 and 1800 (time limit). This portfolio has two different configurations for the optimal and satisficing tracks. For each configuration there are different candidate planners and different times per each planner and all of them come from IPC-2011.

NuCeLaR (Núñez et al., 2014) is a portfolio created in two independent steps. The first one is based on a clustering to split the training problems into groups. Once the set of training problems is split into different subsets (clusters), it computes a different portfolio configuration for each subset using a technique based on Mixed-Integer Programming. This portfolio has three different configurations for the optimal, satisficing and multi-core tracks. For each configuration there are different candidate planners and different times per planner.

ArvandHerd (Valenzano et al., 2014) is a sequential satisficing planner that uses a portfolio consisting of lama and Arvand. This portfolio tries to exploit the strengths of the complementary approaches of random-walk and best-first search based planning used by Arvand. The multi-core version has three threads of different Arvand configurations and one of lama. The single core configurations, first, run the Arvand planner for a defined period of time and then it switches to lama. The agile configuration runs Arvand for 3 minutes and 2 minutes for lama.

Bidirectional Fast Downward (BiFD) portfolio (Alcázar et al., 2014) combines FD progression with FD Regression. The aim of the combination is to exploit the advantages of searching in both directions. This portfolio performs a single preprocessing phase, including the computation of h^2 heuristic in both directions. The time for each search is based on which direction has the least amount of time so far.

USE (Sadraei and Agmadi, 2014) is a sequential portfolio planner that uses forward and backward search. The forward search used is a stochastic planner that takes advantage of "Useful Operator Selection", heuristic functions and preferred operators of lama. When the search process does not go forward, the portfolio resets the search with backward search. This process is repeated until the goal is reached.

Freelunch (Balyo, 2014) consists of two planning algorithms. The first one is a simple heuristic forward search algorithm, and the second is a SAT based approach. The time limit for the forward search is set to 10 seconds, the rest of the time is used for SAT search and post planning optimization.

AllPACA (Malitsky et al., 2014) is a portfolio which automatically chooses which of several planners to run for the planning task that is given. It chooses a planner that is expected to solve the given planning task the quickest based on several features of the planning task. It is a dynamic per instance portfolio that took part in the optimal track. It was designed to select a planner that solves the instance in the shortest time. The predictive model is based on random forest and the input planners are the optimal ones from IPC-2011 with only 65 features extracted from PDDL and SAT formalisms.

ASAP (Vallati et al., 2014) is a domain-dependent static portfolio using an additional representation of the planning problem with macro-actions and outer and inner entanglements. The offline learning of this portfolio consists of select the best encoding in a domain in a specific planner. To this end, all combinations of their learning procedure are executed. The final version is made up of the best combination of the planning representation and the planner works with it until a solution is found. This portfolio presents two versions, one optimized by quality and other one by time. AGAP (Vallati et al., 2014) is an improved version of ASAP. It is an automatic algorithm selection approach for planning that, for a given domain, initially learns additional knowledge, in the form of macro-operators and entanglements. The knowledge is used for creating a different encoding of the given planning domain and problems and selects the most promising algorithm for optimizing the quality of the solution plans.

IBaCoP, IBaCoP2 (Cenamor et al., 2016) and LIBaCoP2 (Cenamor et al., 2014b) are the portfolios as the result of this thesis. IBaCoP is the Pareto-QT selection, IBaCoP2 is the previous one and the classification model and LIBaCoP2 is IBaCoP2 and the regression model.

The list of planning portfolios would not be complete without IBaCoP, IBaCoP2 and LIBaCoP2. Since they are the object of this thesis, we will not give more details on them until the next section.

The previous described portfolios with a summary of the characteristics of the Table 2.1 appears in the Table 2.2.

Table 2.2: Summary of the state-of-the-art planning portfolios. For each portfolio appears: number of planners (N), if there is any criteria to select the planners (F), time strategy (T), ordering strategy (S), if the planners share information between components (Pl), if present additional information of Knowledge (K) and the submitted track (TK).

	N	F	T	O	C	Pl	K	TK
BUS	6	no	Round Robin	Confidence	Dynamic per Problem	no	no	Agile
FDSS-2011	15-6	yes	Lowest Maintain the score	Performance + Contribution	Static	yes	no	Satisficing, Optimal
FDSS-2014	27	yes	Lowest Maintain the score	Performance + Contribution	Static	yes	no	Satisficing
FD-Uniform	21	yes	ET	–	Static	yes	no	Satisficing, Optimal, Agile
FD-Cedalion	8	yes	SMAC	–	Static	yes	no	Satisficing, Optimal, Agile, Learning
MiPlan	12	yes	Optimal	Less time	Static	no	no	Satisficing, Optimal, Multi-core, Learning
DPMPLAN	14	yes	Optimal	Less time	Static	yes	no	Satisficing, Optimal
NuCelar	21	yes	Optimal	Less time	Dynamic per Problem	no	no	Satisficing, Optimal, Multi-core
ArvandHerd	4	no	Arbitrary	Different proportions	Static	yes	no	Satisficing, Agile, Multi-core
BiFD	2	no	Depend of the problem	Proportions	Static	no	no	Satisficing
Freelunch	2	no	Arbitrary	Fix	Process of the system	yes	no	Satisficing
USE	2	no	Depend of the problem	Proportions	Static	no	no	Satisficing, Agile, Multi-core
AllPACA	12	yes	All for one	Confidence	Dynamic	yes	no	Optimal
PbP	7-8	no	Round Robin	Macros	Dynamic per Domain	no	yes	Learning
AGAP	5	yes	All for one	Learning	Dynamic per Domain	yes	yes	Learning
IBaCoP	11	no	ET	–	Static	yes	no	Satisficing, Agile, Multi-core
IBaCoP2	11	yes	ET	Confidence	Dynamic	yes	no	Satisficing, Agile, Multi-core, Learning
LIBaCoP2	11	yes	Predictive Model	Confidence	Dynamic	yes	no	Learning

2.7 Empirical Performance Modelling in AI Solvers

The idea of exploiting the synergy of different solvers to improve the performance of the individual ones is applied in propositional satisfiability problems (SAT), constraints satisfiability problems (CSP), answer set programming (ASP) and in the scope of this dissertation, AP. The SAT area involves considerable research into the importance of selecting the components of the portfolio (Xu et al., 2012a) and how select each component automatically (Lindauer et al., 2015b). The study of strategy selection in this area also includes per-instance selections (Lindauer et al., 2015a). Furthermore, there is an intensive study into the solver's run time prediction (Hutter et al., 2015), including a good characterization of the satisfiability task. In other scopes of the AI, CSP has portfolio configurations based on machine learning techniques such as SUNNY (Amadini et al., 2014b) and other empirical research (Amadini et al., 2014a). For example in ASP, the ASP-based Solver Scheduling (Hoos et al., 2012) is a multi-criteria optimization problem and provides the corresponding ASP encoding. We follow with the main approximations in Automated Planning.

Definition 10 (Empirical Performance Model (EPM))**.** Empirical Performance Models are created to predict the performance of algorithms on previously unseen input, including previously unseen problem instances, previously untested parameter settings, or both.

Roberts et al. (Roberts et al., 2008) presented a set of learned models to assess a set of features, set of planners, and the search space topology. This study demonstrated that having more features could improve accuracy in the predictive models. Additionally, they examine the models to get planner dependencies and to identify problem with a similar structure. This research continues (Roberts and Howe, 2009) to show that models learned from the planners performance on known benchmarks up to 2008. They include a study into the empirical planner's performance, it consist of a set of 28 planners on 4,726 problems. They limited their study to STRIPS and ADL problems and their problem characterization is defined by 19-32 features extracted from the domain and problem definitions. This study includes predictive models based on the success and time of each planner, in which the time models are less accurate than the success ones. This research incorporates a study into the feebleness of the existing benchmark problems and how improve them.

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 relationship between the topology under delete relaxation heuristics and the causal graph together with as DTGs. The feature extraction process is built on top of the FF planner (Hoffmann and Nebel, 2001). This system distinguishes between an overall analysis and a local analysis. The overall analysis shows the absence of local minima once and for all, for the entire state space of a given planning task. Local analysis computes what we call the success rate, which estimates the percentage of individual sample states not on local minima and thus allows finer distinctions to be made. Finally, a diagnosis summarizes structural reasons for the analysis failure, thus indicating domain aspects that may cause local minima. Calculating these features is, however, computationally expensive (Fawcett et al., 2014).

SATzilla (Xu et al., 2012b, 2008) is a system that extracts a group of 138 features from the propositional satisfiability (SAT) problem and these features are used to predict the runtime of solvers. Since a pre-processing step can significantly reduce the size of the CNF formula (especially in industrial-like instances), they chose to apply the preprocessing procedure on all instances first, and then to compute instance features on the preprocessed instances. These can be categorized as problem-size features, graph-based features, balance features, proximity to horn formula features, DPLL probing features, and local search probing features. This group of features is proposed to characterize a SAT task and it is used to find a solver in a portfolio approach for SAT.

Recently, other research (Fawcett et al., 2014) has generated models for accurately predicting the planning run time. These models exploit a large set of instance features, including many of the features detailed in Section 4.3. These features are derived from the PDDL and SAS$^+$ representations of the problem, a SAT encoding of the planning problem and short runs of the planners. Some other features are extracted using 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. Continuing this research (Rizzini et al., 2015), they introduced four dynamic portfolio approximations per optimal planning and 5 different static strategies. These portfolios are used all of the optimal submitted planners at IPC-2014 and the domains that were used in the optimal track in the same IPC, but not in the same problems. They generated 200 problems with the same distribution as in the IPC.

3

Planner Filtering Methods: A Multi-criteria approach

The planner community creates more efficient planners every year. Most of these planners are available for improving and developing better systems. Therefore, they could be used as base planners in a new portfolio. However, using all the planners in a portfolio is an impractical idea, since it requires time that increases linearly with the number of base planners. Discarding the idea of having all planners in a portfolio, the portfolio should be created according to the construction method seen in the previous chapter. This procedure requires all selected planners with the benchmark set to be executed. As a consequence, if the number of selected planners is large, this training phase will be computationally expensive. For this reason, it is necessary to select a small group of planners. A first idea for selecting the planners is to create a ranking with the IPC metrics as time, quality or coverage. Nevertheless, using these metrics in a ranking might not be helpful because it could create portfolios with planners that are not complementary: they all are good but solve the same problems and fail in the same ones. This selection process should give us a group of significantly different planners based on different criteria.

For this purpose, we want this filtering process to select a diverse, but small, subset of planners which have few elements to be distributed in the available execution time. In this chapter, we present a ranking criteria based on a multi-objective technique, taking into account previous metrics (time and quality).

The chapter is organized as follows. First, we describe the metrics and the scope to select the planners. Then, we present our proposal, a multi-criteria approximation. Finally, we show the results of the planner filtering method, the planner selection per metric and the results of the portfolio configured using the previous planner selection mechanism.

3.1 Filtering Methods Based on Quality, Time and Coverage

The planner filtering process consists of the pre-selection of suitable candidate base planners from a larger number of available planners. There is sufficient evidence that there is no overall best planner across a variety of benchmarks. However, it can be verified empirically that some planners dominate over others. Consequently, it does not make sense to include, those that are always worse in terms of performance metrics as base planners. Some of these metrics were described in Section 2.4, like **coverage**, **time** and **quality**. So a first approach to the planner filtering problem could be to compute previous metrics and make decisions based on these values. A simple approach is to use the metrics to create planner rankings, and then, to select the top planners in accordance with this ranking. In the literature, there are examples that follow this approximation. For example, FDSS (Helmert et al., 2011) uses the selection of planners that maximizes the score (It might not be meant to select the individually best ones; it uses the ones that complement each other best), whilst MIPLAN (Núñez et al., 2015a) uses the portfolio configuration that obtains the best achievable performance in terms of quality (in a overall optimization).

3.1.1 Scope of the metrics

To make a ranking, we need to define the scope of this ranking. We have identified three different levels, **problem**, **domain**, and **overall** (See Figure 3.1).

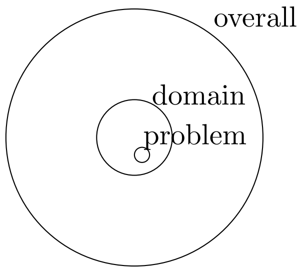


Figure 3.1: Scope of planning ranking selection

Definition 11 (Problem level)**.** Given a problem p, a metric m, and a set of planners Pl, a problem-based ranking $R_p^m(Pl)$ is defined as the result of ordering each planner by the score obtained with the given metric m on the planning problem p: $R_p^m(Pl) = \{pl_1, \ldots, pl_n\}$, such that $m(pl_i, p) \leq m(pl_{i+1}, p)$.

The VBS is created based on a problem level scope. This portfolio always has the best planner per problem in accordance with one metric, where a planner pl_b is selected when $m(pl_b) = MAX\{m(pl_1, p), \ldots, m(pl_n, p)\}$ for any problem p. The principal drawback of a problem-level scope is the number of resulting planners. If you want to create the VBS, you need to select all the best planners but, as will be shown in

Subsection 3.1.2, this method does not guarantee that the size of the planner set will be reduced. The next scope is a domain level that consists of a group of problems in the same domain.

Definition 12 (Domain level)**.** Given a set of problems P_D in a same domain D, $P_D = \{p_1, \dots, p_n\}$, a metric m, a set of planners Pl, a domain based ranking $R^m_D(Pl)$ is defined as the result of ordering each planner by the score obtained with the given metric m on all problems in the domain D: $R^m_D(Pl) = \{pl_1, \dots, pl_j\}$, such that $m(pl_i, P_D) \leq m(pl_{i+1}, P_D)$.

This approximation generates only one ranking per domain. Therefore, if we only get the top planner in each domain, this method can reduce the number of planners more drastically (in IPC, each domain gathers 20-50 problems in IPC-2006-2011[1]), and it could reject a planner even if it is a best planner per problem. Thus, it is not possible to obtain a VBS; and consequently, it does not work perfectly in a training phase. Nevertheless, the objective of planning selection is to find a group of potential planners that might generalize for future problems.

The last scope is the overall level that consists of generating only one ranking for all problems in all domains.

Definition 13 (Overall level)**.** Given a set of domains G where $G = \{D_1, \dots, D_m\}$, a set of problems in these domains, $P_G = P_{D_1,} \cup \cdots \cup P_{D_n}$, a metric m, a set of planners Pl, an overall-based ranking $R^m_G(Pl)$ is defined as the result of ordering each planner by the score obtained with the given metric m on all planning problems and domains G: $R^m_G(\mathcal{P}) = \{pl_1, \dots, pl_j\}$, such that $m(pl_i, P_G) \leq m(pl_{i+1}, P_G)$.

The last method only generates one ranking for all of the problems in any domain. Selecting only the one top planner in a configuration does not represent a suitable strategy for selecting planners in a portfolio: it is only a criteria for selecting a good planner in certain settings, as will be shown below. It is important to highlight that the benchmark suite also influences this process. On the one hand, a small set of problems/domains could create a non-generalizable planner selection. If a benchmark suite is inappropriate, the selection of the planners could be inefficient and it might imply that the final portfolio does not solve new problems. Additionally, if the training benchmark is too small, introducing a new instance or a group of them will trigger changes in the ranking, independently of the scope level. Conversely, a large benchmark set could solve the volatility and inefficient problems, as the training set should be sufficient to create a more stable ranking.

For example, an example of a small training benchmark is made up 5 of problems from 5 domains and a set of 502 initial planners. This planner selection includes 5 planners in the problem and domain scope and 1 planner at the overall level. This planner selection is volatile because if a new problem is introduced the planner selection will change in most of the cases. The ranking presents 5 planners with a score of between 0 to 5, but on average the planners have 2 points. This is easy when introducing a new problem (a possible new point in the score), as the ranking changes, regardless of

[1]Other domains gather from 5-150 problems.

the problem corresponding to a new domain or not. This situation is totally different when the training set includes 500 problems from 10 domains and the same set of 502 initial planners. In the first case, the number of selected planners is different, a maximum number of 500 planners at the problem level, 10 planners at the domain level and 1 planner at the overall level. This planner selection is more stablr than previous one. If a new problem is introduced the probability of changing the ranking is much lower than the first ranking. The overall level only changes the planner if the difference between the first and the second planner is less than one point, and the second planner gets the best result for the new problem. The domain level presents two possibilities for changing the ranking, the first one is that the new problem is from an included domain. The ranking changes in the same situation of the overall level, only the selected planner is changed if this domain has a small differences between the first and the second planner. The second is when the problem is a new domain, a new ranking is created. There is a possibility that the selected planner is not included, and the number of planners increases from 10 to 11 (the number of selected planners are the same in the two previews cases). The problem level evaluates a new ranking for this problem and selects one planner that could be included in the previous selection. This situation would increase the number of planners from 500 to 501.

In conclusion, a small training set will drastically modify the planner selection at all scope levels. However, an appropriate training set modifies the planner selection with less probability provided that $Probability(R_{problem}) \geq Probability(R_{Domain}) \geq Probability(R_{Global})$, where $Probability(R_x)$ is the probability of changing the ranking at the different levels.

In any case, the next subsection presents an evaluation that provides an intuition as to how the problem level scope will work in general to filter a set of planners.

3.1.2 Empirical Evaluation of Problem Level based Filtering

To carry out this evaluation, we consider a large and diverse group of planners: 27 planners that include the Sequential Satisficing Track of IPC-2011 plus LPG-TD (Gerevini et al., 2006). Although LGP-TD did not compete in IPC-2011, we considered it worthwhile to include it because it is still considered a state-of-the-art planner due to its outstanding performance in previous competitions. Furthermore, this set of planners was the most up-to-date when this dissertation started. To carry out this evaluation, we present the Hypothesis of Increasing Planner Selection. It proposes that if you have a group of potential planners with a large enough set of benchmarks, the final planner selection will tend to include all initial planners.

Hypothesis of Increasing Planner Selection. Given a set of planners Pl and a set of n problems $P = \{p_1, \ldots, p_n\}$, where each planner pl_i has a probability $P(p_i)$ such that $\epsilon \geq P(p_i) > 0$ to obtain the best solution for some problem, the final selection of planners, Pl_{end} goes towards Pl when n increases.

In other words, this hypothesis says that, as we introduce more and more problems, an instance-based ranking will provide every planner some chance of being selected since there is a chance greater than zero (ϵ) that it is the best for a particular problem.

To illustrate this hypothesis empirically, we show two different examples; the first one is the domain *openstacks* with its 20 problems, and the second one is an *openstacks* and *parcprinter* domain with 20 problems each (both domains from sequential satisficing track IPC-2011). The *openstacks* domain is based on the "minimum maximum simultaneous open stacks" combinatorial optimization problem [1] . *Parcprinter* domain models the operation of the multi-engine printer. Given a set of heterogeneous works: color or mono, one-sided or two-sided print. The domain tries to solve the planning task to optimize the printers according to requirements.

These examples are evaluated with a quality score because it is the most standard metric in the planning community. For each evaluation, all best planners are shown because the quality score metric does not include any criteria for solving a tiebreak. The second part of this evaluation includes all IPC-2011 domains to evaluate the minimum number of planners that achieve the VBS with different metrics (coverage, quality and time). This case includes a criterion to solve a technical draw.

The results of the first evaluation with 20 *openstacks* domain problems is shown in Figure 3.2 with a quality score. For the 20 problems, 13 different planners achieve the best solution in at least one problem. In several cases, several planners obtain the best quality, so all of them could be selected. This is not a normal situation and, for example, problems 5 and 6 only select one best planner. Nevertheless, other problems such as 1, 4 and 7 select three planners. In conclusion, there is no general rule for determining the number of best planners per problem.

Figure 3.3 details one domain in red and the other in blue. This example consists of 20 *openstacks* problems like the previous figure (Figure 3.2) and 20 *parcprinter* problems. This situation gets worse with more problems from the second domain. These results only discard 5 planners from the initial 28, with only 2 domains. It is important to highlight that every competition presents more than 5 domains, and these results only show a small sample of the filtering process that should be included in a planner filtering process. This technique is not enough to select an appropriate planner selection. The figures show that these initial planners are suitable, and this strategy is not enough to obtain a proper planner filtering. This selection means that we are getting close to all initial planners with these problems (26 of the 28). This planner filtering does not consider planners that have no probability of achieving a best result from a single problem, at least in this training set.

This method only excludes useless planners because this approximation does not represent the minimum set of planners that achieve the best score in this group of problems.

The next experiment is carried out from 0 to 14 domains from IPC-2011 (280 problems in total). This approximation presents a "tie-break" rule to determine the minimum number of planners that achieve VBS according to a metric and it illustrates how the number of planners increase as the number of problems/domains increase. The X axis represents the increase in the number of domains in the selection criteria (20 problems per domain).

Figure 3.4 details the planners necessary to obtain the minimum number of planners to reach an approximation of VBS, following the three different metrics, coverage, quality and time. The minimum number of planners for the coverage criterion is only

[1]The problem is that a manufacturer has a number of orders, each for a combination of different products, and can only make one product at a time.

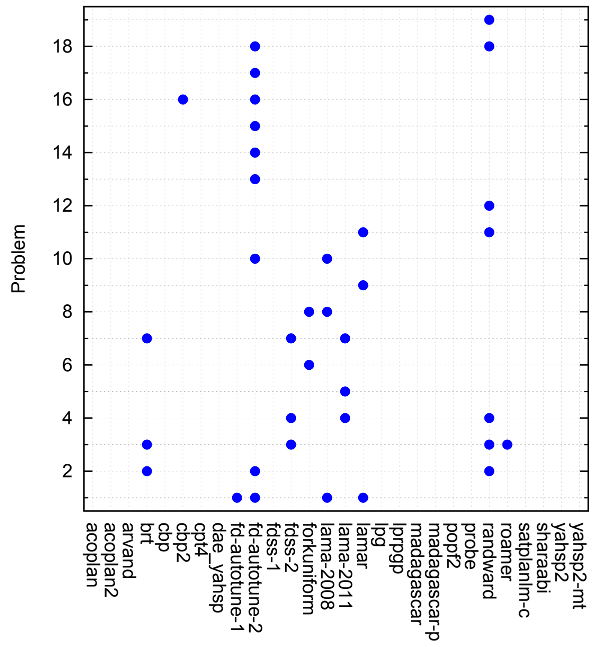


Figure 3.2: Best planner per problem in terms of quality score at *Openstacks* domain. Axis x is the problem and axis y is the initial set of planners. Each row could select one or more planners in function of whether the planner achieves the best solution.

three, which means that with only three planners we can solve all problems in all the domains. However, this method does not guarantee the best quality or the minimum planning time. With the quality criterion, 17 of the 28 initial planners are selected, so this mechanism discards 11 planners. This planner filtering could suppose an initial appropriate sub-set. With the last metric, time, 21 planners are selected, so only 7 planners are discarded.

These results show a big difference between the three metrics. The selected planners are different in the case of the time or quality score (this point will be shown in Figure 3.5). These planners do not match because they are built with different objectives. The "time" planners focus on solving problems as fast as possible, whereas that the "quality" planners focus on achieving the best solution, without taking into account how much time takes. This situation raises two problems; the first one is the limitation in selecting the minimum number of planners following one criterion. This method discards many planners, several of them are potential candidates, however they are not selected because one of them achieved similar results in training phase (remember that this method only includes one planner in the case of a tie). The second problem is the disjunction of the selected planners with another criterion. The best planner for time might not be the best planner for quality. These metrics are tangential, because in a normal situation better quality implies more time. For this reason, a planner filtering

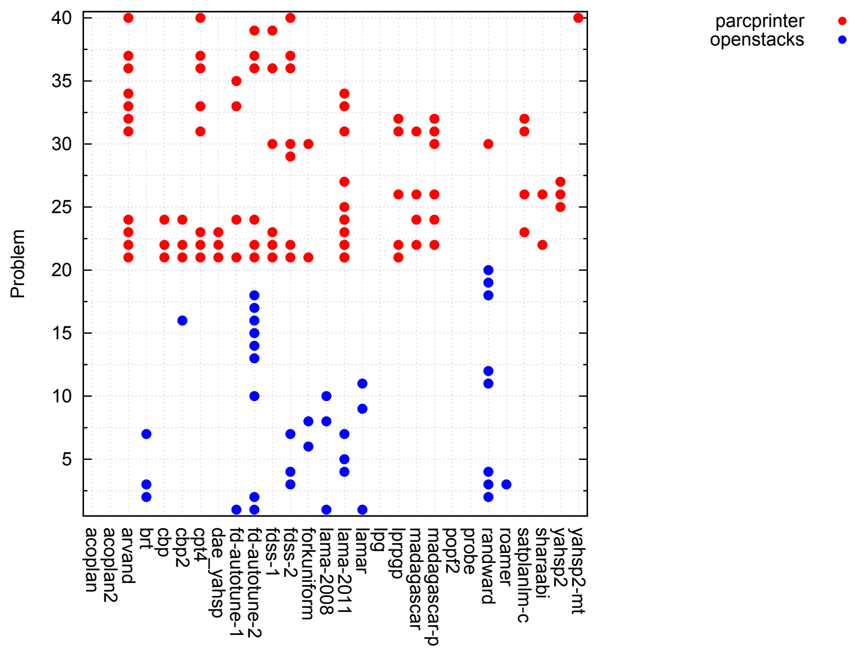


Figure 3.3: Best planner per problem in terms of quality score in *Openstacks* and *Parcprinter* domains. x-axis is the problem and y-axis is the initial set of planners. Each row could select one or more planner in function on whether the planner achieves the best solution. This evaluation presents 40 problems in 2 domains.

criteria should include adequate suitable set of diverse planners, not only focusing on one criterion.

Figure 3.5 represents a figurative separation between planners, where there are three different groups: planners that are good at coverage, the planners that work very fast and planners that achieve the best qualities. If a planner achieves the best results on quality means that the planner is also good at solving problems, because these metrics are correlated. This is the same situation as with the time metric: if a planner solves the planning task in less time than the others, it means that the planner has solved it. Both metrics (quality and time) are correlated to coverage, but they are not among them.

3.2 Pareto Dominance-Based Planner Filtering

Two main conclusions arise from previous section. On one hand, a planner selection approach should be performed at the domain level, since the problem level does not generate accurate planner sets. On the other hand, it is important to consider both metrics, quality and time, in a planner filtering method. To address these issues, we propose a multi-criteria planner filtering method carried out at the domain level. This

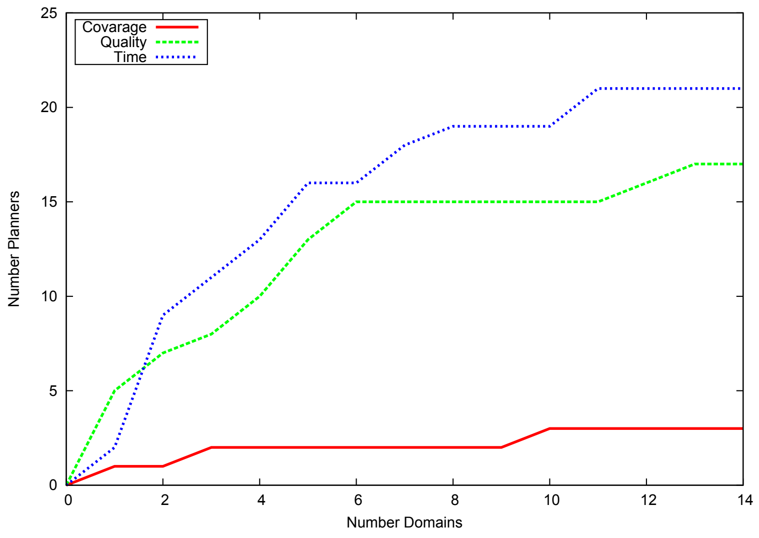


Figure 3.4: Number of Selected planners by different metrics (Quality, Time and Coverage) at 14 sequential satisficing domains (IPC-2011).

multi criteria planner filtering is described in terms of Pareto dominance, described in definition 14.

Definition 14 (Pareto dominance)**.** Pareto dominance is a metric that determines whether at least one individual is better off without making any other individual worse off, given a certain initial allocation of goods among a set of individuals. Given two points $p_1 = \langle x, y \rangle$ and $p_2 = \langle x', y' \rangle$, p_1 dominates p_2 when $x \geq x'$ and $y > y'$ or $x > x'$ and $y \geq y'$. Another case, there is no dominance relationship between these points.

For filtering, we propose to run the candidate planners on a representative set of benchmarks and then evaluate them in terms of time and quality. To consider both

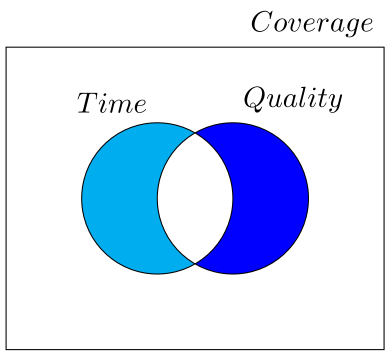


Figure 3.5: Coverage, time and quality planner sets

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 et al., 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, *pl*, the dominance relationships between p and the rest of planners are computed as described in Definition 15.

Definition 15 (QT-Pareto dominance)**.** A QT-Pareto dominance is the determination of the best planner with a QT metric; where a planner pl_1 gets a tuple $\langle Q, T\rangle$ in a problem p, and a planner pl_2, in the same problem, gets $\langle Q^{'}, T^{'}\rangle$. The planner pl_1 dominates pl_2 if $Q \geq Q^{'}$ and $T < T^{'}$.

In our approximation, a tuple $\langle Q, T\rangle$ Pareto dominates the tuple $\langle Q', T'\rangle$ if and only if $Q \geq Q'$ and $T < T'$. Therefore, we do not admit that a planner dominates another with less time but less quality.

Finally, the QT-Pareto score for a domain is the sum of the points achieved in all of the problems in the domain following Definition 16. The idea of this selection mechanism is as follows: if a planner demonstrates a 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 those planners with the maximum QT-Pareto score for at least one domain. We refer to this procedure as *QT-Pareto Score Filtering.*

Definition 16 (QT-Pareto Score)**.** QT-Pareto Score is the metric for discovering which planner dominates the others; where, a planner pl_1 gets $\frac{N}{N^{*}}$ points, where N is the number of tuples where pl_1 Pareto-dominates another planner, and N^{*} is the number of different tuples in which planner *pl* appears.

3.3 Evaluation of Planner Filtering Approaches

In this section, we carry out a complete evaluation of the planner filtering approaches, which is the basis for deciding which planners should be included in a future portfolio. To select the candidate planners, we propose four different planner filtering metrics (coverage, time, quality and Pareto QT) at a domain level. The benchmarks for computing the planner filtering is the set of domains and problems of the sequential satisficing track of IPC-2011 (Linares López et al., 2015). At the end of the section, a first set of portfolios with all the planners pre-selected by each of the metrics studied are evaluated.

Table 3.1: Quality and Time planner filtering per domain. For each metric, the selected planners, score and coverage are shown.

	Quality			Time		
Domain	Q	Coverage	Planners	T	Coverage	Planners
pegsol	20.00	20	ARVAND, LAMA-2011	18,97	20	MADAGACAR-P
scanalyzer	18.53	20	ARVAND	20.00	19	YAHSP2
parcprinter	19.42	20	ARVAND	15.34	20	MADAGACAR-P
openstacks	19.09	20	FD-AUTOTUNE-2	16.92	19	LPRPGP
tidybot	16.67	19	LAMAR	14.27	18	PROBE
nomystery	18.97	19	ARVAND	17.15	15	MADAGACAR-P
woodworking	19.99	20	FDSS-1	13.91	20	FORKUNIFORM
sokoban	18.57	20	FD-AUTOTUNE-1	17.25	19	FDSS-2
visitall	19.71	20	DAE_YAHSP	13.31	20	YAHSP2-MT
transport	16.72	19	ROAMER	16.10	20	YAHSP2
elevators	18.01	20	FD-AUTOTUNE-2	17.39	20	LAMA-2011
parking	18.11	20	LAMA-2011	17.43	20	LAMAR
barman	19.37	20	FD-AUTOTUNE-1	18.59	20	PROBE
floortile	12.00	12	LPG-TD	09.37	12	LPG-TD
total	255.15	269		225.99	265	

3.3.1 Planner Filtering

Table 3.1 shows the planners selected by the quality (Q) planner filtering and the time (T) planner filtering.

In this table, the score obtained per domain and the total for all the planner selection methods are set out (each column is evaluated by its metric since both metrics are not comparable). In addition, each metric includes the coverage per domain and the problems solved with the perfect portfolio in each planner filtering. The sub-set of planners for Q are LAMA-2011, ARVAND, FD-AUTOTUNE-2, LAMAR, FDSS-1, FD-AUTOTUNE-1, DAE_YAHSP, ROAMER and LPG-TD (9 planners from 14 domains). ARVAND is selected by four domains, FD-AUTOTUNE-1 by two as the same as LAMA-2011 and FD-AUTOTUNE-2. The T planner filtering presents other planners: MADAGACAR-P, YAHSP2, YAHSP2-MT, LPRPGP, PROBE, FORKUNIFORM, FDSS-2, LAMA-2011, LAMAR, and LPG-TD. There are 10 planners in each group, and there are only three planners included in both.

Table 3.2 shows the planner filtering in terms of coverage. In this case, the planner's column shows all those planners that solve the maximum number of problems per domain. The big difference with the previous table is that now, more than 2 planners get the maximum performance in 11 domains; only tidybot, nomystery and floortile reveal a small set of planners. Therefore, the selected planners, in this case, are 22: LPG-TD, LAMA-2011, FDSS-2, PROBE, FDSS-1, FD-AUTOTUNE-1, ROAMER, FORKUNIFORM, LAMAR, FD-AUTOTUNE-2, ARVAND, LAMA-2008, RANDWARD, LPGRGP, MADAGASCAR, MADAGASCAR-P, ARVAND, BRT, CBP, CBP2, DAE_YAHSP, YAHSP2 and YAHSP2-MT.

Table 3.2: Coverage planner filtering with the planners that obtain the maximum coverage per domain.

Domain	Coverage	Planners
pegsol	20	LAMA-2011,FDSS-2, PROBE, FDSS-1, FD-AUTOTUNE-1, ROAMER, FORKUNIFORM, LAMAR, FD-AUTOTUNE-2, ARVAND, LAMA-2008, RANDWARD, LPRPGP, MADAGASCAR-P
scanalyzer	20	LAMA-2011, FDSS-2, PROBE, FDSS-1, FD-AUTOTUNE-1, ROAMER, LAMAR, ARVAND, LAMA-2008, BRT
parcprinter	20	LAMA-2011, FDSS-2, FDSS-1, FD-AUTOTUNE-1, FORKUNIFORM, ARVAND, YAHSP2, MADAGASCAR-P
openstacks	20	LAMA-2011, FD-AUTOTUNE-1, ROAMER, LAMAR, FD-AUTOTUNE-2, ARVAND, LAMA-2008, RANDWARD, CBP2, CBP
woodworking	20	LAMA-2011, FDSS-2, PROBE, FDSS-1, FD-AUTOTUNE-1, ROAMER, FORKUNIFORM, LAMAR, RANDWARD
transport	20	PROBE, YAHSP2, YAHSP2-MT, CBP2, DAE_YAHSP
tidybot	19	LAMAR, BRT
elevators	20	LAMA-2011, FDSS-2, PROBE, FDSS-1, FD-AUTOTUNE-1, FORKUNIFORM, ARVAND, BRT
visitall	20	LAMA-2011, PROBE, LAMA-2008, RANDWARD, YAHSP2, YAHSP2-MT, DAE_YAHSP
nomystery	19	FD-AUTOTUNE-2, ARVAND
parking	20	LAMA-2011, FDSS-2, FDSS-1, LAMAR, LAMA-2008, RANDWARD
sokoban	19	LAMA-2011, FDSS-2, FDSS-1, FD-AUTOTUNE-1
barman	20	LAMA-2011, PROBE, FD-AUTOTUNE-1
floortile	12	LPG-TD
total	269	

This planner filtering only discards a few planners of the initial set (6 planners), and most from them came from the last positions in the ranking results at IPC-2011.

Furthermore, we propose a multi-criteria QT-Pareto Score Filtering to reduce the initial set of candidate planners as an alternative technique to planner filtering. This technique integrates both quality and time in a multi-criteria style, as described above. Table 3.3 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 between both values. The QT-Pareto score values closer to 20 reflect that the planner is able to beat the other planners at most problems. PROBE was the best planner in 4 domains. However the other planners only stood out in one domain. This reinforces the need to find a diverse subset of planners. Finally, out of 28 initial planners, the QT-Pareto score filtering pre-selected the subset of 11 planners as candidate 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.

Table 3.3: List of the best planners ordered by their QT-Pareto score for each domain of IPC-2011.

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

3.3.2 Planner Selection

Planner filtering based on QT-metric has three planners in common with *Q* and *T* filtering: LAMA-2011, LAMAR and LPG-TD. The selected planners are different depending on the criteria, for example at *Q* filtering has ARVAND, FD-AUTOTUNE-1 and FD-AUTOTUNE-2 and *T* filtering has YAHSP2-MT, FDSS-2 and PROBE. The QT-Pareto Filtering has only one planner. Madagascar that does not appear in both techniques (However, MADAGASCAR-P is a modificated version of it.)

Table 3.4 shows the ranking of planners following the quality score of the IPC results (Linares López et al., 2015) and which of them were selected by Q, T, C and QT-Pareto Score Filtering [1]. It is worth noting that 12 of the 13 best planners in the IPC are built on top of FD, which reduces the diversity of the planners. The planner filtering strategies select several of them: in the C case, the 20 best planners according to the ranking plus LPG-td are selected. On the other hand, the Q filtering only selects 9 planners and the selection criterion does not look at ranking selection, and 7 of 9 are based on FD. The T filtering is the technique that selects fewer planners based on top of FD (5 planners) although it has one planner more than the Q filtering. The QT-Pareto Score Filtering only includes 8 planners built on top of FD and one planner more than T filtering. It should be pointed out that the last three selections of the QT-Pareto Score Filtering are planners from the lower positions in the table which, as will be demonstrated later, increases the diversity of the portfolio and its performance.

It is important to highlight that there are three planners that are included in the four planner filtering approaches: LAMA-2011, LPG-TD and LAMAR, so it seems that these planners are always good options for creating a portfolio. However, it is important to remember that the selection is made by domain. For example, the selection of LPG-TD is due to the floortile domain, where this planner solved more problems than the others. This is not the case for the other planners, as shown in Table 3.2, and LAMA-2011 and LAMAR obtains the maximum number of problems solved at least in 5 domains, but they are also good in some domains in terms of quality or time.

[1]This results does not correspond with the IPC-2011 results because LPG-TD were not included, for this ranking we evaluate the planners included it.

Table 3.4: List of planners ordered by its score at IPC-2011. The columns *Q*, *T*, *C* and *QT* indicates whether the planner is eligible by the Quality, Time, Coverage and QT-Pareto Score Filtering, respectively. The last column shows whether the planners are built on the top of FD. The planners are not selected by any strategy are not include in this table.

Ranking	planner	Q	T	C	QT	FD
1	LAMA-2011	√	√	√	√	√
2	FDSS-1	√		√		√
3	FDSS-2		√	√	√	√
4	FD-AUTOTUNE-1	√		√	√	√
5	ROAMER	√		√		√
6	FORKUNIFORM		√	√		√
7	FD-AUTOTUNE-2	√		√	√	√
8	PROBE		√	√	√	
9	ARVAND	√		√	√	√
10	LAMA-2008			√	√	√
11	LAMAR	√	√	√	√	√
12	RANDWARD			√		√
13	BRT			√		√
14	CBP2			√		
15	DAE_YAHSP	√		√		
16	YAHSP2		√	√		
17	YAHSP2-MT		√	√	√	
18	CBP			√		
19	LPRPGP		√	√		
20	MADAGASCAR-P		√	√		
22	MADAGASCAR				√	
24	LPG-TD	√	√	√	√	
Total	28	9	10	22	11	12

3.3.3 Planner Filtering Portfolio

In this section, we report a first approximation on the creation of planning portfolios, built on the aforementioned results. Specifically, we have built the portfolios with the planners selected by each filtering approach described. The resulting portfolios have a uniform configuration (assign the same time per each candidate planner) and are evaluated with the domains at IPC-2014. The results of each planner filtering are shown in terms of quality and coverage. Furthermore, we included a baseline strategy to compare, *Overall Equal Time* (OET). This strategy is a non-informed strategy which does not carry out any planner filtering, and assigns equal time for each available planner. Given that we have 28 planners (all the participants in 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. The QT portfolio gives 163 seconds per planner, Q gives 200 seconds, T gives 180 and C gives 81.

Table 3.5 shows that the results in terms of quality. They show the QT is the best over the rest of the techniques obtaining more than 56 points of difference. The quality results in this table are essentially related to the coverage results. Furthermore, we present Figure 3.6 which depicts the problems solved in $1,800$ seconds. The QT solves 249, Q 191, T 185 and C 178. QT solves at least 58 problems more than the other strategies.

The best planner at IPC-2011, LAMA-2011, solves 166 problems, less than the base strategy OET. Somehow, it is normal that every portfolio in this section achieves better solutions than the best individual planners, since all configurations have this planner as a component. These results confirm the initial idea of the thesis that "the use of planning portfolios could improve the performance of every single planner."

In conclusion, we can say that our approximation, QT-Pareto Score Filtering, works better than the other planner filtering strategies in our training data set. Furthermore, from the results we can derive some insights as regards the different configurations. The difference in score between OET and QT reveals the importance of making a pre-selection of candidate planners 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 demonstrate that it is more important to maintain diversity than to increase running time per planner. For instance, C planner filtering obtains similar results to those using the original 28 (OET) at the end of the running time. These portfolios solve the same number of problems (178) but C planner filtering gets 0.65 points more in terms of quality. This is a logical result because the C planner filtering only discards 6 planners from the initial 28. The T planner filtering solves 185 problems with 10 planners, so the results increase by 3.78% from no applying any filter or by applying coverage filtering. The Q planner filtering solves 191 problems with 9 planners, so this filtering increases the performance by 3.14% with respect to the T planner filtering. These results demonstrate that using more planners does not mean better performance. This approximation has better results than T planner

Table 3.5: Results in terms of quality of all planner filtering strategies and all initial planner components. QT is the portfolio created from the QT-Pareto Score Filtering, Q is the portfolio from the quality selection, T is the portfolio with time selection and OET is the portfolio with all the initial planners. All portfolios follow a uniform distribution of time to all planners.

Portfolio	QT	Q	T	C	OET
Number planners	11	9	10	22	29
Hiking	19.14	**19.38**	18.56	19.12	18.17
Barman	**19.64**	17.65	19.14	19.38	16.74
Thoughtful	**19.54**	18.79	18.53	18.61	14.51
GED	19.17	18.52	**19.29**	19.08	18.28
Openstacks	19.66	**19.99**	19.50	14.88	15.44
Parking	18.99	**19.00**	16.99	9.72	17.64
Maintenance	15.53	**16.84**	13.89	16.46	15.00
Tetris	15.22	**15.89**	7.38	12.51	4.99
CityCar	**13.50**	12.69	7.82	8.68	5.99
Visitall	**16.90**	9.02	9.12	3.94	13.25
Childsnack	**18.73**	5.37	8.24	7.53	11.95
Transport	**19.95**	5.98	5.40	5.69	8.92
Floortile	**17.00**	3.43	1.88	3.43	4.81
CaveDiving	6.39	0.00	**7.00**	**7.00**	0.00
total	**239.35**	182.56	172.73	166.03	165.68

filtering with one planner less. The last approximation, *QT-Pareto* Filtering, solves 249 problems, increasing the performance by 23.29% from the Q planner filtering.

In the next chapter, we only consider the *QT-Pareto* based planner selection described in this chapter. See more details of the planners in Appendix A.

3.4 Summary

There are a lot of planners in the state of the art and it is not possible to include all them in a portfolio. One of the most successful techniques uses a combination of search and heuristics, however this approximation is dependent on the system. Other techniques use a sequential combination of planners that are selected arbitrarily or by a filtering criteria based on IPC metrics. This chapter proposes "scope of the metrics" to perform the planner filtering to limit the number of selected planners. This is an idea that has been never formalized before, but somehow extended to AP. We carried out an empirical evaluation based on the *Hypothesis of Increasing Planner Selection*, in which we presented the idea of the selection per problem not being successful in terms of planner filtering. We solved this problem by using the domain level in the filtering criteria. Notwithstanding, this obstacle being solved, there is another issue to resolve, the metric of the filtering criteria. The metrics contemplated are tangential and, it is not easy to select a diverse set of planners. We proposed a multi-criteria method

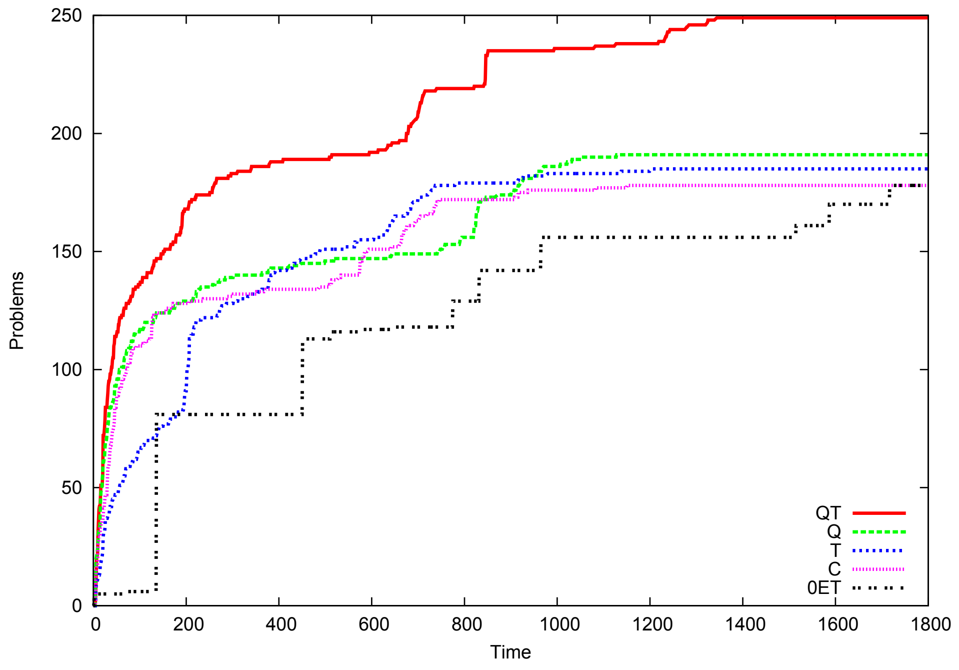


Figure 3.6: Solved problems (Coverage) of all planner filtering strategies using uniform time per each candidate planner.

to obtain a good set of base planners. The results of our proposal against the single criterion achieve promising results.

4

Planning Problem Characterization and Empirical Performance Modeling

One of the objectives of this thesis is to define a group of features able to characterize a planning problem. Initial representations (Roberts and Howe, 2009) were poor and not enough to determine every planning task properly, as explained in Section 2.7. These characterizations were based on the PDDL formalism and mainly consisted of domain differentiation and problem-size prediction. Moreover, there are other representations that are not based on the PDDL, such as SAT formulation (Kautz et al., 2006). These features are based on the propositional satisfiability formula (Xu et al., 2008) and are computed in the procedure to simplify this representation to solve the planning task in a SAT solver. We propose to improve the current representation of the planning problem based only on planning formalism, so we create new ones based on SAS$^+$ formulation, landmarks, fact balance and, by taking advantage of the heuristic functions and the translation process from PDDL to SAS$^+$. These features could be created under a general Data Mining process to produce predictive models. In this chapter we demonstrate that, with the new features, the characterization of the planning task improves in terms of prediction accuracy. Therefore, they are suitable for creating empirical models of the planner's behavior.

The chapter is organized as follows. First, we explain a general process for creating empirical performance models using a new set of features extracted from different planning paradigms. Then, we describe all the features used in this process. After that, we present the final datasets, and a feature selection process. Finally, we present the results from classification and regression perspective.

4.1 Planning Performance Modeling Process

As stated above, modelling the planner's behavior as a function of the 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 4.1. 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

,

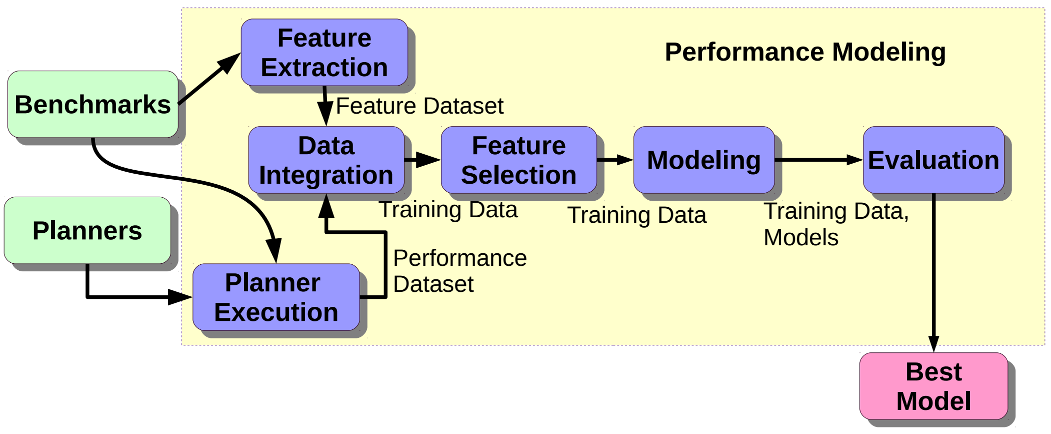


Figure 4.1: General Diagram for Learning the Planning Performance Predictive Models

The complete set of features is listed and organized by their category in the next section. The Data Integration process receives the features and the performance datasets as inputs to produce a final dataset in accordance with the modeling goal. In the dataset for the classification task, a training/test instance includes the planning task features plus the planner's 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, in addition, in most cases this time is unknown. A training/test instance in the regression dataset includes the planning task features, the planner's name and the time this planner takes to find the 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 might degrade the modeling capabilities of some learning techniques (Blum and 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 can 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 et al., 2011, Witten and Frank, 2005), which will reflect the generalization ability of the different models when making predictions on unseen data.

The *evaluation* is an integral part of the development process of the model. It helps to find the best model that represents our data and to evaluate how well the model chosen will work in the future. There are several methods of evaluating models (Cenamor et al., 2013): cross-validation, leave-one-domain-out, split evaluation. All techniques use a test set (not seen by the model) to evaluate the model performance.

- Cross-validation (Browne, 2000): this evaluation permits the classification accuracy (percentage of times that the model outputs the expected class) of a classifier to be estimated in the future, or the predicting capability (relative absolute error of the predicted value in respect to that expected) of a regression model. A cross-validation splits the data randomly into k groups, $(k-1)$ used for training the model and the rest to test the model learnt. 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.

- Split evaluation: this evaluation splits the available data into two sets: a training set and a test set. For example, the problems are split in two sets depending on its identifier: odd or even. There is not bad choice because the problems used in 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.

- Leave-one-domain-out: this evaluation permits how the models will behave in problems of unseen domains to be evaluated. The approach is based on the leave-one-out evaluation method, which in machine learning can be seen as a cross-validation, where k is set to the amount of available data. This method is a cross-validation in which the data is not separated in folds randomly, but per domain. 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 models learnt in previously unseen planning domains.

4.2 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, we will always refer to the satisficing tracks of the competitions. The included domain and problems appear in Table 4.1.

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. We obtained a total of $1,251$ planning tasks.

Table 4.1: List of domains used to generate the training set. This information includes the track, year, name of the domain and number of problems per domain.

Track	Domains	Problems
seq-sat-2006	openstacks, pathways, tpp, trucks	30
	rovers, storage	40
seq-sat-2008	pipesworld	50
	openstacks-adl	31
	cybersec, openstacks, pegsol, scanalyzer, sokoban,transport, woodworking, elevators	30
seq-sat-2011	barman, elevators, floortile, nomystery, openstacks, parcprinter, parking, pegsol, scanalyzer, sokoban, tidybot, transport, visitall, woodworking	20
learning-2008	gold-miner, matching-bw, n-puzzle, parking, sokoban, thoughtful	30
learning-2011	barman, blocksworld, depots, gripper, parking, rovers, satellite, spanner, tpp	30

4.3 Feature Extraction

The first step in 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, in 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. According to the mechanism for generating these features, we classify them into the following categories: PDDL, FD instantiation, SAS$^+$, Heuristic, Fact Balance (FB) and Landmark (Summary in Table 4.2).

Next, we describe the features in detail.

4.3.1 PDDL Features

These basic features, which appear in Table 4.3, offer information on the domain and the size of the problems. In fact, most problem generators receive the number of objetcts of each type and the number of goals as input, so they can determine the size of the instance. However, these basic features and many others that can be extracted from the domain definition will not be sufficient to discriminate between instances of the same size. There are 8 features in this section.

Table 4.2: Features Summary, the first column is the type, the second is the number of features per type and the last one refers to whether the features are extracted from the FD system and, if so, what step.

Type	Number	FD
PDDL	8	$\surd$ - translation
FD Instantiation	16	$\surd$ - translation & preprocess
SAS$^+$	50	$\surd$ - preprocess
Heuristic	16	$\surd$ - search
FB	10	$\times$
Landmark	14	$\surd$ - search
Total	114	5/6

Table 4.3: PDDL Features.

Name	Description
Objects	Number of objects in the problem.
Goals	Number of goals in the problem.
Init	Number of in facts in the initial state.
Types	Number of types in the domain.
Actions	Number of actions in the domain.
Predicates	Number of predicates in the domain.
Axioms	Number of axioms in the domain.
Functions	Number of functions in the domain.

4.3.2 FD Instantiation Features

A Fast-Downward pre-processor instantiates and translates the planning tasks into a finite domain representation (Helmert, 2009). From the output of this process, 16 features are obtained, as shown in Table 4.4.

4.3.3 SAS$^+$ Features

Features based on a finite domain representation of SAS$^+$ has an associated *Causal Graph* (CG) and a set of *Domain Transition Graphs* (DTGs), as described in section 2.3.2. 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. All these features are described in tables 4.5 and 4.6 for the CG and 4.7 for the DTG. There are a total of 50 features in this section.

Table 4.4: Extracted Features from console output in the FD system.

Name	Description
Relevant facts	Number of facts marked as relevant by FF instantiation.
Cost metric	Whether action costs are used or not.
Generated rules	Number of created rules in the translation process to create SAS$^+$ task.
Relevant atoms	Number of relevant atoms found in the translator process.
Auxiliary atoms	Number of auxiliary atoms found in the translator process.
Final queue length	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.
Total queue pushes	Number of times an element has been pushed into the queue.
Implied effects removed	Number of implied effects removed. Where the implied effects that the translator knows are already included.
Effect preconditions added	Number of implied effects added.
Translator variables	Number of created variables in SAS$^+$ formulation.
Derived variables	Number of state variables that correspond to derived predicates or to other artificial variables not directly affected by operator applications.
Translator facts	Number of facts that the pre-process takes into account.
Mutex groups	Number of mutex groups.
Total mutex size	The sum of all mutex group sizes.
Translator operators	Number of instantiated operators in SAS$^+$ formulation.
Total task size	The allowed memory for the translation process.

Table 4.5: CG Features obtained from SAS$^+$ representation I.

Name	Description
	General Features
Variables	Number of variables of the CG.
HV Variables	Number of high-level variables.
Total Edges	Number of edges.
Total Weight	The sum of the edge weights.
	CG Ratios
VE Ratio	Ratio between the total number of variables and the total number of edges. This ratio shows the level of connection in the CG.
WE Ratio	Ratio between the sum of the weights and Number of edges. This ratio shows the average weight for the edges.
WV Ratio	Ratio between the sum of the weights and the number of variables.
HV Ratio	Ratio between 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
Input Edge	Maximum, average and standard deviation of the number of incoming edges for each variable. (3)
Input Weight	Maximum, average and standard deviation of the sum of the weights of the incoming edges for each variable. (3)
Output Edge	Maximum, average and standard deviation of the number of outgoing edges for each variable. (3)
Output Weight	Maximum, average and standard deviation of the sum of the weights of the incoming edges for each variable. (3)

Table 4.6: HV Features obtained from SAS$^+$ representation II.

Name	Description
	Statistics of high-level Variables (HV)
Input Edge	Number of incoming edges for each of the high level variables. This value produces three new features following the same computation as *Input Edge CG.* (3)
Input Weight	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 *Input Weight CG.* (3)
Output Edge	Number of outgoing edges for each of the high level variables. (3)
Output Weight	The sum of the weights of the incoming edges for each high level variables. (3)

Table 4.7: DTG Features obtained from SAS$^+$ representation.

Name	Description
	General Aggregated Features DTG
Vertices	The sum of the number of nodes of all DTGs.
Edges	The sum of the number of edges of all DTGs.
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
edVa Ratio	Ratio between the total number of edges and the total numbers of variables. This ratio shows the level of connection in the DTG.
weEd Ratio	Ratio between the sum of the weights and the number of edges.
weVa Ratio	Ratio between the sum of the weights and the number of variables.
	Statistics of DTGs[1]
Input Edge	Maximum, average and standard deviation of the number of incoming edges for a vertex in a DTG. (3)
Input Weight	Maximum, average and standard deviation of the sum of the weights of the incoming edges of all nodes.
Output Edge	Maximum, average and standard deviation of the number of outgoing edges for a vertex in a DTG. (3)
Output Weight	Maximum, average and standard deviation of the sum of the weights of the outgoing edges of all nodes. (3)

4.3.4 Heuristic Features

Some features are based on the heuristic value of the initial state, computed from a set of widely-used unit cost heuristic functions. These heuristics can be obtained at a reasonable cost as we will show in Table 4.12. To guarantee a domain-independent estimation, the heuristics are computed with a unit cost that helps in the characterization of the problem size and/or difficulty. The features derived from heuristics are described in tables 4.8 and 4.9. There are 16 features in this section.

Table 4.8: Heuristics features with unit cost included.

Name	Description
Max (Bonet and Geffner, 2000, Bonet et al., 1997)	The maximum of the accumulated costs of the paths to the goal propositions in the relaxed problem.
Landmark cut (Helmert and Domshlak, 2009)	The sum of the costs of each disjunctive action landmark that represents a cut in a justification graph towards the goal propositions.
Landmark count (Richter et al., 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.
FF (Hoffmann and Nebel, 2001)	The cost of a plan that reaches the goals in the relaxed problem that ignores negative interactions.
Additive (Bonet and Geffner, 2000, Bonet et al., 1997)	The sum of the accumulated costs of the paths to the goal propositions in the relaxed problem.
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.
Context-enhanced additive (Helmert and Geffner, 2008)	The causal graph heuristic modified to use pivots that define contexts relevant to the heuristic computation.
Red-black (Katz and Hoffmann, 2013, Katz et al., 2013)	It is an approach to partial delete relaxation, where red variables take the relaxed semantics (accumulating their values), while black variables take the regular semantics.

Table 4.9: Features Related with the Red-Black heuristic.

Name	Description
Black Variables	Number of black variables in the Red-Black compilation.
Black Root Variables	Number of black root variables in the Red-Black compilation.
Red Black Variables	Number of variables in the Red-Black compilation.
pairs values connected	Number of variables with all pairs of values connected in the Red-Black compilation.
Connected goal	Number of variables with all values connected to goal in the Red-Black compilation.
Strongly parents	Number of black variables with strongly connected only with the parents in the Red-Black compilation.
Maximal effects	Maximal number of side effects for black variables in the Red-Black compilation.

4.3.5 Fact Balance Features

Some features are based on the relaxed plan (RP) of the initial state, extracted when computing the h^{FF} heuristic. In this section, we describe the general background and process to extract them. Remember, a planning task $\Pi = \langle \mathcal{S}, \mathcal{O}, c, s_0, s_\star \rangle$, where each $o \in \mathcal{O}$ is defined as the tuple $\langle pre(o), add(o), del(o) \rangle$ representing the action preconditions, added effects and deleted effects respectively. A plan π is a sequence of actions that achieve the goals $s_\star$.

Definition 17 (Relaxed plan)**.** A **relaxed plan** π^+ is the sequence of actions that solves a relaxed planning task, i.e. a task in which the deleted effects of the actions have been ignored (Hoffmann and Nebel, 2001).

The plan $\pi^\pm$ is defined as the sequence of actions corresponding to π^+, but with the actions taken from the original task. This plan is obviously not applicable in most cases, but it provides useful information for a variety of search control techniques, such as look-ahead states (Vidal, 2004). For the fact balance, $\pi^\pm$ is the base for encoding additional information that is not incorporated into the heuristic values of delete relaxation heuristics. Specifically, the information is computed from the relaxed plan of the initial state, which is denoted by $\pi^\pm_{s_0}$. We will still call $\pi^\pm$ a relaxed plan to highlight that it is basically obtained by the original procedure of π^+.

Definition 18 (The balance of a fact)**.** Given a relaxed plan $\pi^\pm$, **the balance of a fact** p is a function $\mathcal{B}$ that computes the difference in the number of times p is added and the number of times p is deleted in $\pi^\pm$.

$$\mathcal{B}(p, \pi^\pm) = |\{o | o \in \pi^\pm \wedge p \in add(o)\}| - |\{o | o \in \pi^\pm \wedge p \in del(o)\}| \quad (4.1)$$

The idea behind computing the balance of facts is that we can reflect with a number whether the relaxation is altering the possible application of a sequence of actions more or less from $\pi^{\pm}$. A negative balance for a fact p indicates that $\pi^{\pm}$ will probably not be applicable, and p will have to be recovered through other actions in the real plan. The intuition behind fact balances is that high positive values would characterize the (relaxed) problems for a given domain easier, since achieved facts do not need to be deleted many times.

Furthermore, the *Relaxed Planning Graph* (RPG) built by the FF provides additional information.

Definition 19 (Relaxed Planning Graph)**.** An RPG is a sequence of proposition and action layers $P_0, A_0, P_1, A_1, \ldots, A_{t-1}, P_t$, representing the reachability analysis of the relaxed planning task. Each action in $\pi^{\pm}$ has an associated layer that we denote with $level(a)$.

The value of the h^{FF} heuristic is the number of actions in the relaxed plan. However, this number does not reflect many properties of the structure in $\pi^{\pm}$ and its corresponding RPG. The idea behind these new features is to partially encode this underlying structure, and therefore find the difference between tasks that may have the same heuristic value in the initial state.

Definition 20 (Level balance of a fact)**.** A **level balance of a fact** p **at layer** l is a function $\mathcal{B}_{[l]}$ that computes the difference of the number of times p is added and the number of times p is deleted by actions of $\pi^{\pm}$ that belongs to layer $l-1$.

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

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

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

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

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

```
$fp^+ = 0; fp^- = 0; dist\_fp = 0$
for $p \in \mathcal{P}$ do
    if $is\_goal(p)$ then
        target = 1
    else
        target = 0
    if $\mathcal{B}_{[l]}(p, \pi^{\pm}) \neq 0$ then
        if $\mathcal{B}_{[*l]}(p, \pi^{\pm}) \geq target$ then
            $fp^+ += \mathcal{B}_{[*l]}(p, \pi^{\pm})$ - target
        else
            $fp^- +=$ target - $\mathcal{B}_{[*l]}(p, \pi^{\pm})$
            lgap = 1
            diff = $\mathcal{B}_{[*l]}(p, \pi^{\pm})$
            while (lgap +l) < layers(RPG) and diff $\geq$ target do
                diff += $\mathcal{B}_{[l]}(p, \pi^{\pm})$
                lgap += 1
            $dist\_fp$ += (target - $\mathcal{B}_{[*l]}(p, \pi^{\pm})$) * $2^{\text{lgap}}$
```

The balance footprints computed by the algorithm in Algorithm 1 are associated to particular layers of RPG. Furthermore, their magnitude is affected by the number of actions in the previous layer. Therefore, to compute the balance and unbalance features in a normalized way, we aggregate the value of each layer by multiplying it by a weight that represents the proportion of actions that appear in each particular layer of RPG. The distortion measure involves information of several layers, therefore it will not be weighted. There are 10 features in this section and we summarize them in Table 4.10

4.3.6 Landmark Features

Features based on the landmark graph detailed in Section 2.5.2 are also computed. To try to maximize the dependence between the different results of the landmark graph, the features are extracted through the merging of the landmarks and orderings from h^m with $m = 1$, Richter, Helmert and Westphal (RHW) and Zhu/Givan landmarks. The specific features are listed in Table 4.11. There are 14 features in this section. This landmark graph is obtained by FD using the Merged Landmarks in the pre-processing part.

Table 4.10: Features about the fact balance.

Name	Description
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. $(\mathcal{B}(p, \pi^{\pm}_{s_0}), \forall p \in \mathcal{S})$. (3)
RP_goal	Minimum, average and variance of the number of times that a goal is deleted in the computation of the relaxed plan. $(\mathcal{B}(g, \pi^{\pm}_{s_0}), \forall g \in s_\star)$(3)
Ratio_ff	Ratio between the value of the max and FF heuristic. This proportion shows the idea of parallelization of the relaxed plan.
RP Balance Ratio	Aggregate the value of each layer multiplying it by a weight that represents the proportion of actions that appear in each particular layer of the occurrences in which a fact has a positive balance. $\sum_{i=1}^{layers(RPG)} \frac{\lvert A_{i\;1}\rvert}{\lvert A\rvert} \times fp_i^+$
RP Unbalance Ratio	Aggregate the value of each layer multiplying it by a weight that represents the proportion of actions that appear in each particular layer of the occurrences in which a fact has a negative balance. $\sum_{i=1}^{layers(RPG)} \frac{\lvert A_{i\;1}\rvert}{\lvert A\rvert} \times fp_i^-$
Balance Distorsion	Aggregate the value of each layer for the distorsion of unbalanced facts. $\sum_{i=1}^{layers(RPG)} dist\_fp_i$

Table 4.11: Features about the landmark graph.

Name	Description
n_landmarks	Number of landmarks included in the merged landmark graph.
Number Edges	Number of edges in the landmark graph.
edVa ratio	The ratio between the edges and landmarks.
Father nodes	Number of nodes that do not have incoming edges in the landmark graph.
Children nodes	Number of nodes that do not have outgoing edges in the landmark graph
Between nodes	Nodes in the graph that have father and children.
Input Edges	Average, maximum and standard deviation of the input edges of all nodes in the landmark graph. (3)
Output Edges	Average, maximum and standard deviation of the output edges of all nodes in the landmark graph. (3)

4.3.7 Process & Time to extract features

The 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 and landmark features are computed using the Fast-Downward search engine, and the 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, features that are not computed and missing values are assumed.

However, there are different difficulties in the extraction of the features. The PDDL features are easier to extract than the SAS$^+$ ones. The extraction of the FD Features has the same complexity as the SAS$^+$ ones and they all are extracted in the same procedure. Finally, the heuristic and landmark values have the same complexity as the latter, but they can only be extracted if the pre-processing of the FD is completed. It is worth noting that the inclusion of these features in the FD pre-processor is useful in the next steps, because FD-based planners will take advantage of this part of the process: these phases in the planners are built on top of FD and are skipped in the running process. Several features are obtained out of the FD pre-processor, as Heuristics, Fact Balance and landmark features. Table 4.12 has an additional row indicating the extra time that the feature extraction process takes and that is not included in any other part.

Table 4.12 also shows the success rate for extracting the features of each type from the training problems, and the average, standard deviation, maximum and median 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 table shows the time in which the features are calculated in the whole process. The time required to compute the heuristic features is only the time for calculating the value in the initial state, but these features are also needed to finish the FD pre-process. The average time to extract all of the features are close to 24 seconds: it is a small part of the execution of the planner since it only spends 1.3% of the total planning time (i.e. 1800 seconds). In this sense, the most expensive part is the computation of the heuristic features: it takes 13 seconds to extract them and they are not always extracted (with only an 87.54% success rate). However, the process does not fail when the features are not extracted, and the system replaces these features with a missing value. This method guarantees the same number of input features, regardless of whether they were correctly extracted. Furthermore, this procedure ensures a perfect working in the deployment phase.

4.4 Data Integration

Once the features are defined, we are ready to define the instances generated. The performance data comprises 13,761 instances (i.e., 1,251 problems × 11 planners) where 9,156 correspond to successfully solved problems, and 4,605 to planning failures.

Table 4.12: Summary of the extracted features with the average, standard deviation, maximum and median time in seconds (s.) required to extract them. These processes are on the top of the two first steps of all planners based on FD.

Process	Success	Mean (s.)	SD	Max (s.)	Median (s.)
Tranlate (PDDL)	97%	5.98	14.42	50.71	0.36
Preprocess (FD & SAS$^+$)	97%	1.10	5.00	50.02	0.06
Fact Balance	93%	0.73	4.54	50.18	0.03
Heuristics	87.54%	13.15	33.14	230.36	0.68
Landmarks	87.54%	1.72	7.96	100.06	0.24
Mercury	97%	0.01	0.00	0.02	0.00
Extra time		0.44	0.55	4.29	0.22
Total		23.11	-	485.63	1.60

Figure 4.2 shows the proportion of instances solved by each candidate planner. These results are a summary of the entire training phase (see more details in Appendix C).

To generate this dataset, the execution of all candidate planners previously selected in the Chapter 3 (Section 3.3.3) is required. Each candidate planner takes 1,800 seconds for each planning task (2,251,800 seconds per planner).

The first dataset corresponds to a binary classification problem. An instance is positive if the planner finds at least one solution, and negative if the planner does not solve the problem in 1,800 seconds.

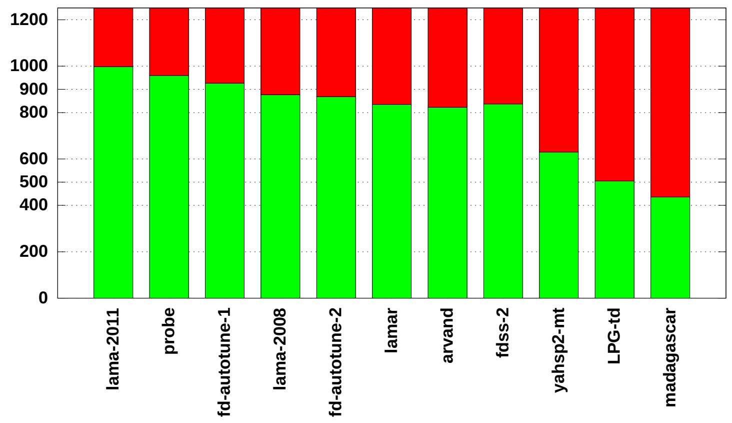


Figure 4.2: Proportion of solved (green) and unsolved (red) planning tasks in the training instances for each of the candidate planners.

A second dataset is created by filtering "False" instances from the previous dataset, and it only preserves positive values. This dataset has an output attribute that represents the time to get the best solution in all cases.

In summary, we have two different datasets:

1. **Classification dataset**: Input Features (Section 4.3) + Planner + True/False (114 numeric input attributes, 1 enumerate input attribute and 1 enumerate output attribute).

2. **Regression dataset**: Input Features (Section 4.3) + Planner + Time, where $0 < Time < 1800$ (114 numeric input attributes, 1 enumerate input attribute and 1 numeric output attribute).

4.5 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 modeling purposes. 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 the most relevant features for characterizing a planning task.

The selection of attributes is important because it can mean the difference between successfully and meaningfully modeling the problem and not. This feature selection can identify potential problems such as dominant attributes, usefulness or attributes with a high percentage of missing values. There are three general automated methods of feature selection algorithms: filter methods, wrapper methods and embedded methods.

- Filter Methods: apply a statistical measurement to assign a score to each feature. The features are ranked by the score and either selected to be kept or removed from the dataset. The methods are often univariate and consider the feature independently, or with regard to the dependent variable.

- Wrapper Methods: consider the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A predictive model is used to evaluate a combination of features and assign a score based on the accuracy of the model.

- Embedded Methods: learning which features the best contribution to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods are regularization methods.

We carried out an a informal evaluation using the Weka (Witten and Frank, 2005) toolkit. The main algorithms do not provide a suitable feature selection. This process almost gave a selection which only included the planner and the output attribute. The models generated gave us information on the planner's performance in the training phase. These methods did not work and we carried put a manual filter based on a decision tree. The feature selection was carried out using the J48 algorithm, a top-down induction algorithm for building decision trees (Quinlan, 1993), by selecting the features that appear in the top nodes of the tree (Grabczewski and Jankowski, 2005). Decision trees make an implicit feature selection as the model includes queries to those features considered relevant. Therefore the root node of a decision tree is the most important feature from the classification point of view, since it has the maximum attribute gain (Quinlan, 1993). The feature selected in each node of the tree is the best one for the instance space in that node, i.e. the set of instances that satisfies the conditions from the root to that node. The attribute with the highest gain ratio is selected as the splitting attribute. The non-leaf nodes of the decision tree are considered as relevant attributes.

After applying this feature selection process on the feature dataset, the total number of features decreased from 114 to 14. This leads to a reduction in the size of the dataset of around 87%. Table 4.13 contains the list of features resulting from the feature selection process. The selection chooses features from almost every category. We kept both datasets separately for the modeling and evaluation process, one with all available features (f-all) and the other one with the selected features (f-14).

Table 4.13: List of features from the feature selection.

Type	Features
PDDL	types
	actions
FD Instantiation	generated rules
	effect conditions simplified
	translator variables
SAS^+	avg. edge CG
	variables DTG
	avg. input edge DTG
FB	relevant facts
	avg. fact balance
	unbalance ratio
	balance distorsion
Landmark	number edges
	edges / variables ratio

In tables 4.14 and 4.15 in the next section, the differences in performance of the prediction models learned with all of the features and the reduced set are shown. The models used to select the features were built with the J48 algorithm using all of the features and by pruning. This mechanism is a sensitive instrument that permits avoiding overfitting by pruning the decision tree.

4.6 Classification Models

The detailed performances of the predictive models are evaluated with a 10-fold cross-validation on a uniform random permutation of all training data. This is an evaluation technique that guarantees the best trade-off between predictive quality and complexity of data (Friedman et al., 2001). It is very extensive in the literature (Fawcett et al., 2014, Hutter et al., 2015, Roberts and Howe, 2009).

We have trained the classifiers using 27 classification algorithms provided by Weka (Witten and Frank, 2005), which includes different types of model such as decision trees, rules, support vector machines and instance-based learning. We recall that training instances include the planning task features described plus the name of the planner and the Boolean feature indicating whether this planner solved the planning task or not.

Table 4.14 shows the accuracy, standard deviation and area under the ROC curve of each model when evaluated using a 10-fold cross-validation. The ZeroR algorithm predicts the mode (majority class) of the training data. This dummy classifier serves as baseline to show that most of the models are able to obtain a high degree of accuracy when predicting out-of-sample data. For both datasets f-all and f-14, the best model is that generated using Rotation Forest (Rodriguez et al., 2006). Rotation Forest produces an ensemble of classifiers (decision trees) in which each base classifier is trained on different feature sets. To create the training data for a base classifier, the whole feature set is randomly split into K subsets and Principal Component Analysis (PCA) is applied to each subset. All of the main components are retained in order to preserve the variability information in the data. Thus, K-axis rotations take place to form the new features for a base classifier.

The accuracy of the best model (90.50%) is far from the accuracy of the default model (ZeroR). 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 the features.

Seven algorithms have a statistically better accuracy with the f-14 dataset and twelve of them have the similar accuracy, but in all cases they were close to the results with all features.

4.7 Regression Models

We have trained regression models with only the positive instances of the classification training phase. In the classification phase, all of the planners have the same proportion of instances, but in this case, not all of the planners have the same number of instances given that they solved a different number of problems.We have trained the models with 18 regression algorithms, also provided by Weka.

Table 4.15 shows the results of the *Relative Absolute Error* (RAE) and the correlation coefficient obtained by each algorithm. In the table, the small RAE values are better.

The feature selection achieved better results in only four algorithms, and only one worse than the initial dataset. All of these cases are close to the best performance with all features. The rest of algorithms achieved comparable results with and without feature selection.

The best algorithm for f-all is Decision Table (Kohavi, 1995). We have selected the Decision Table model for the regression task in both datasets (f-all and f-14). In

Table 4.14: Accuracy, standard deviation and Area under curve ROC 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

Algorithm	**f-all dataset**				**f-14 dataset**				
	acc.	±	SD	AUROC	*acc.*	±	SD	AUROC	
rules.ZeroR	66.54	±	0.0	0.50	66.54	±	0.0	0.50	
rules.Ridor	81.49	±	2.4	0.74	80.82	±	2.3	0.74	
rules.PART	86.88	±	0.9	0.92	86.37	±	1.0	0.92	
rules.JRip	85.74	±	1.1	0.83	85.23	±	1.1	0.83	
rules.DecisionTable	86.48	±	1.0	0.92	86.41	±	1.2	0.92	
trees.REPTree	86.19	±	0.9	0.90	85.73	±	1.0	0.90	
trees.RandomTree	82.28	±	1.6	0.83	84.04	±	1.0	0.85	○
trees.RandomForest	84.90	±	1.3	0.91	86.12	±	0.9	0.92	○
trees.J48	86.46	±	0.9	0.90	86.62	±	0.9	0.90	
trees.DecisionStump	66.83	±	0.72	0.63	66.83	±	0.79	0.63	
lazy.LWL	66.91	±	0.8	0.73	66.83	±	0.8	0.76	
lazy.IBk K 1	76.26	±	1.2	0.73	77.92	±	1.0	0.77	○
lazy.IBk K 3	78.82	±	1.1	0.81	80.98	±	1.0	0.83	○
lazy.IBk K 5	78.10	±	1.1	0.83	81.09	±	1.0	0.85	○
meta.RotationForest	**90.50**	±	0.8	0.96	**90.07**	±	0.8	0.96	●
meta.AttributeSelectedClassifier	86.63	±	0.92	0.90	86.69	±	0.90	0.90	
meta.Bagging	87.41	±	0.88	0.93	86.97	±	0.86	0.93	●
meta.ClassificationViaClustering	52.08	±	1.88	0.53	61.66	±	5.21	0.57	○
meta.ClassificationViaRegression	88.46	±	1.00	0.94	87.70	±	0.93	0.94	●
meta.MultiClassClassifier	78.03	±	0.89	0.84	72.36	±	1.01	0.77	●
functions.SimpleLogistic	76.00	±	0.89	0.82	72.37	±	0.97	0.77	●
bayes.NaiveBayes	68.48	±	1.08	0.71	68.08	±	0.97	0.72	
bayes.NaiveBayesUpdateable	68.48	±	1.08	0.71	68.08	±	0.97	0.72	
bayes.BayesNet	69.01	±	1.18	0.76	73.64	±	1.20	0.79	○
functions.SMO	76.97	±	1.00	0.71	72.65	±	1.00	0.65	●
functions.MultilayerPerceptron	86.33	±	1.71	0.90	83.38	±	1.40	0.88	●
functions.RBFNetwork	68.31	±	1.46	0.63	68.37	±	1.07	0.65	

the following sections, the regression model will always refer to that trained with the Decision Table algorithm.

The output in the Planning Performance Modeling Process is the best two models, one for classification, the other for regression. The best models in the first part are the Rotation Forest, and the Decision Table in the second case. These models are used in the next section of the thesis.

4.8 Summary

The characterization of planning tasks used before this Ph.D. thesis is partially unsuitable for differentiating each planning task since it is only based on the PDDL representation. This issue is a big deal because if the problem definition is not correlated to the performance of the planners, a system that uses this information would not work well. To solve these problems, we propose a set of new features that significantly improve the characterization of the problem. These features came from a different planning representation (PDDL and SAS$^+$) and we take advantage of other information that the planners use in the search process (for example: landmarks, heuristics, to name a

Table 4.15: Results for the 10-fold cross-validation in the regression models. *RAE* is the Relative Absolute Error and standard deviation (SD) is the correlation coefficient. Symbols ○, ● means statistically significant improvement or degradation respectively

	f-all dataset			f-14 dataset			
	RAE		SD	RAE		SD	
rules.ZeroR	100.00	±	0.00	100.00	±	0.00	
rules.M5Rules	72.40	±	4.31	71.95	±	2.74	
rules.ConjunctiveRule	90.46	±	1.75	90.11	±	1.85	
rules.DecisionTable	**64.13**	±	3.62	**66.83**	±	3.08	○
trees.DecisionStump	88.86	±	1.55	88.86	±	1.55	
trees.REPTree	73.21	±	2.96	72.44	±	2.91	
trees.M5P	64.69	±	4.31	70.50	±	4.50	
lazy.IBk K 1	87.18	±	5.02	94.06	±	6.79	○
lazy.IBk K 3	84.58	±	3.36	83.56	±	4.57	
lazy.IBk K 5	82.46	±	3.02	80.74	±	3.93	
lazy.KStar	75.36	±	3.25	73.22	±	3.17	●
lazy.LWL	93.57	±	1.24	93.90	±	1.44	
meta.Bagging	69.67	±	2.27	69.18	±	2.25	
meta.AdditiveRegression	84.82	±	2.22	86.94	±	2.29	○
functions.LinearRegression	88.21	±	2.08	90.90	±	2.05	○
functions.LeastMedSq	106.81	±	1.04	106.59	±	0.54	
functions.MultilayerPerceptron	97.65	±	17.68	90.36	±	14.05	
functions.RBFNetwork	95.46	±	2.23	93.15	±	3.35	●

few). This set of features are used in predictive models to learn the performance of the selected planners in the filtering process in Chapter 3. The classification results show a good performance of the models learnt, achieving close to a 90% accuracy and AUROC values of 0.90. All learning algorithms achieve better results than the default model in both metrics. The regression results are not as promising as the classification ones, but they provide some slight knowledge on how predict the planner's runtime. These models are good enough to be used in a portfolio configuration. Furthermore, we show that there is no real difference between f-all and f-14 (using all the features or a selected one respectively). It means that the selected features are enough to obtain similar results, but this selection does not have any advantages in terms of the quality or accuracy in the predictive models learnt.

5

Configuration Strategies to Create Planning Portfolios

Current state-of-the-art portfolio construction methods have different steps, as can be seen in Subsection 2.6.1. The first one included the selection of the candidate planners. Within this first step, the configuration target is another important issue. This configuration target could be *static*, i.e. it is always the same throughout the entire planning problem, or *dynamic*, whose configuration could change planners, execution time and/or order per planning problem or domain. This chapter provides an analysis of most extended techniques, and we propose several to use the empirical performance models that can be learned following the ideas introduced in Chapter 4.

Specifically, in this chapter we review the state of the art in portfolio configurations based on time, order and planner selection. In Section 5.3, we carry out an analysis of the performance of our portfolios according to the number of base planners selected. After that, we propose a set of configurations based on previous studies and an overall algorithm to merge several strategies. Summarizing, we present different ideas to combine previously acquired knowledge, as the predictive models to configure a portfolio per instance.

5.1 Static Strategies

The static portfolios always have the same configuration for all problems and domains. These portfolios always present the same candidate planners with their running time, and the absolute order among them never changes, as described in the next definition.

Definition 21 (Static Strategy)**.** Given a set of planners $Pl = \{pl_1, \ldots, pl_n\}$, a static portfolio is the configuration made up of n pairs, always in the same order and time $Pf = \langle < pl_1, t_1 >, \ldots, < pl_n, t_n > \rangle$, where $\sum_{i=1}^{n} t_i \leq T$, T being the time limit.

However, there is not only one technique to set these variables; for example, FDSS performs a local search in the space of schedules to find a portfolio that maximizes the score of the final portfolio on the training set and assigns the minimum time to solve them for each component. This is one of the possible approximations, there are other approaches. We highlight two different categories, depending on whether we are defining the time assigned or the order of execution. There are many approaches to assign the time to each base planner:

- *Equal Time (ET):* This strategy assigns 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 et al., 2012).
- *Minimum Required Time:* This strategy assigns different times per candidate planner and this time it is the minimum time a planner requires to solve all of the tasks solved by it during the training phase as FDSS.
- *Average Time:* This strategy assigns different times per candidate planner and this time it is the average time that the planners expended in training phase for all of the planning tasks.
- *Maximum Time:* This strategy assigns different times per candidate planner and this time it is the maximum time required to solve every planning task in the training phase.

There is no research into portfolios based on average time or maximum time in static configurations. These strategies are our proposals to assign non-uniform time in these types of portfolio. The main reason for these strategies is based on the idea that more time can be better expended in a planner that introduces a new one if the selected planner obtains the best solution. Assigning more time per planner could guarantee that a problem, which is more complex than the training one, has enough time to be solved. *Minimum Required Time* presents a limitation because the training problems could be smaller, and require less time, than a new test set.

When defining the order, we differentiate:

- *Arbitrary*: this strategy does not have any knowledge on ordering. An example of this ordering could be alphabetical, or appearance.
- *Shorter Time*: this strategy sorts the component solvers of a given portfolio in to the increasing order of the allotted time to run each solver, like the PbP portfolio does.
- *Memory Failures*: this strategy sorts the components according to the fewest memory limit failures, like FDSS.
- *Higher Coverage*: this strategy sorts the components according to the number of problems solved.
- *Density function* this strategy sorts the component according to the contribution of its candidate to the total time, maximizing the area under the curve (Núñez et al., 2015b).

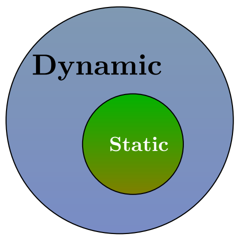


Figure 5.1: A diagrammatic representation of configuration sets and their relationships.

The static strategies are the most extended, but these techniques are less flexible than the dynamic ones. The dynamic strategies are closer to the definition of VBS, because they have the possibility to change the planner to find a better solution. However, these strategies are more complex than the previous ones, and they sometimes require extra knowledge or they become overly expensive (see the scope of strategies in Figure 5.1).

5.2 Dynamic Strategies

These strategies have the ability to change the number of planners, the order and the time assigned to each candidate planner, depending on the problem or the domain, as described in the next definition.

Definition 22 (Dynamic Strategy)**.** Given a problem p, a set of planners $Pl = \{pl_1, \ldots, pl_n\}$ and a maximum time T. A dynamic portfolio is a specific configuration of j pairs such that $Pf = \langle < pl_1, t_1 >, \ldots, < pl_m, t_m > \rangle$ in which, $pl_j \in Pl$ and $\sum_{j=1}^{m} t_j \leq T$.

The only example of portfolios that select different candidate planners appears in the last IPC apart from ours, AllPACA (Malitsky et al., 2014). The closest approximation can be found in SAT solvers, in which they present predictive models to select the best solver per task (Lindauer et al., 2015b, Xu et al., 2012a). These predictive models show the trustworthiness (confidence) of each solver to achieve the solution, in which case, the confidence is an indication of how often the rule has been found to be true. This confidence could be used to select a group of N planners in a portfolio. We describe next a set of configurations in terms of the number or set of planners and the use of the predictive models.

- *Random*: this strategy selects a number N of planners randomly.
- *Classification Model*: this strategy selects the N planners that the classification model returns as true.
- *Best N confidence (BN):* this strategy includes the subset of N planners with the best prediction confidence in the positive class in the portfolio. 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.

- *Logarithmic time:* this strategy includes the subset of N planners with the less predicted run time.
- *Probability distribution:* this strategy includes the subset of N planners if they always have less predicted run time than a fixed time T.

In a classification model, each candidate planner will get a yes/no prediction given a new planning task. The direct use of the boolean variable makes it 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 slice of the total time, in which this assignment can be carried out uniformly or dependently, again, from the predictive models learned.

In terms of time, we differentiate:

- *All static strategies*
- *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.
- *Confidence:* the time is assigned proportionally to the estimated trustworthiness provided by the predictive model.

In terms of order, we differentiate:

- *All static strategies*
- *Random:* this strategy sorts the solvers of the input portfolio randomly.
- *Confidence:* this strategy sorts the solvers with the trustworthiness provided by the classification models in decreasing order.
- *Probability:* this strategy sorts the solvers with the ratio between the probability of success and the time assigned per each candidate planner ($P(p_1)/T(p_1)$) as BUS.

As we have shown, there are many possible strategies for configuring a portfolio, and there is no standard way to determine the best. Furthermore, the number of planners in a portfolio is an unexplored issue. The most extended criterion is to ensure solving the maximum number of problems, or the achievement of the best performance in terms of another metric.

5.3 Estimated Number of Planners

Several strategies need to define “a priori” the number of selected planners, as the *Best N Confidence* strategy, where N must be selected. This is an important decision because there is a trade off between the diversity and effectiveness in the planning resolution. For this reason, we carry out an experimental study that uses the predictive model by changing the number of planners that run in the available time.

In the experiment, the 11 selected planners that result of Section 3.3 are used as initial base planner set. We also use the classification model from Section 4.6 with all the features. The test set is the planning benchmark of sequential satisficing in IPC-2011, the time limit is $1,800$ seconds and the memory limit is 4 Gb. We evaluate different portfolios created with the *Best N Confidence* strategy, where we modify the parameter N. We apply a uniform distribution of time among the selected planners, i.e. $1,800/N$.

Figure 5.2 details the number of problems solved depending on N. These results included two additional configuration to understand the contribution. IBaCoP is the result of the planner filtering applied in Chapter 3 (11 planners). OET is the initial set of planners with uniform time (28 planners). Note, that neither configurations have an initial phase of feature extraction, and they are a sequential execution of all planners. The initial portfolio *OET* solves more problems than a portfolio with only one or two planner selections. Over this limit when the planner selection includes three or four planners, it achieves better results than *OET* but under the portfolio configuration applied by Pareto Filtering (eleven planners). From five to ten, the number of problems solved increase the QT-Pareto Score filtering strategy. The best strategy is 10 planners because it solves 254 problems, however there is not significant difference between the results when N ranges from 6 to 10.

Table 5.1 shows the results of selecting a different number of planners in a portfolio using the predictive model learned in Chapter 4 in the *Best N Confidence* Portfolio Configuration strategy. The name of each column references the N parameter and each cell is the quality obtained in each domain. Additionally the last row presents the number of problems solved. The best results are obtained for $N = 10$, so it only discards one planner from the initial set of 11. However, the portfolio with $n = 7$ obtains the best results in 4 domains. The results of not applying any filtering or the QT-Pareto Score filtering appear in Subsection 3.3.3. These results show that filtering processes improves from 165.68 to 239.35. Even the classification models improve the performance from 6 planners, and with only 5 planners the results are comparable.

Figure 5.3 details the problem’s gain between the planner filtering criteria (IBaCoP) and *Best N Confidence* strategy for different values of N. The value in the graphic is the number of problems solved by the *Best N Confidence* strategy minus the number of problems solved per IBaCoP at different running times. These values could be positive if it is better than IBaCoP, negative if it is under IBaCoP, and zero when there is no difference in the number of problems solved.

All strategies give worse results than IBaCoP until 17 seconds from the execution because these strategies need to run the feature extraction phase. After that, the first strategy that obtains better results is for $N = 8$, close to $N = 7$. In addition, we

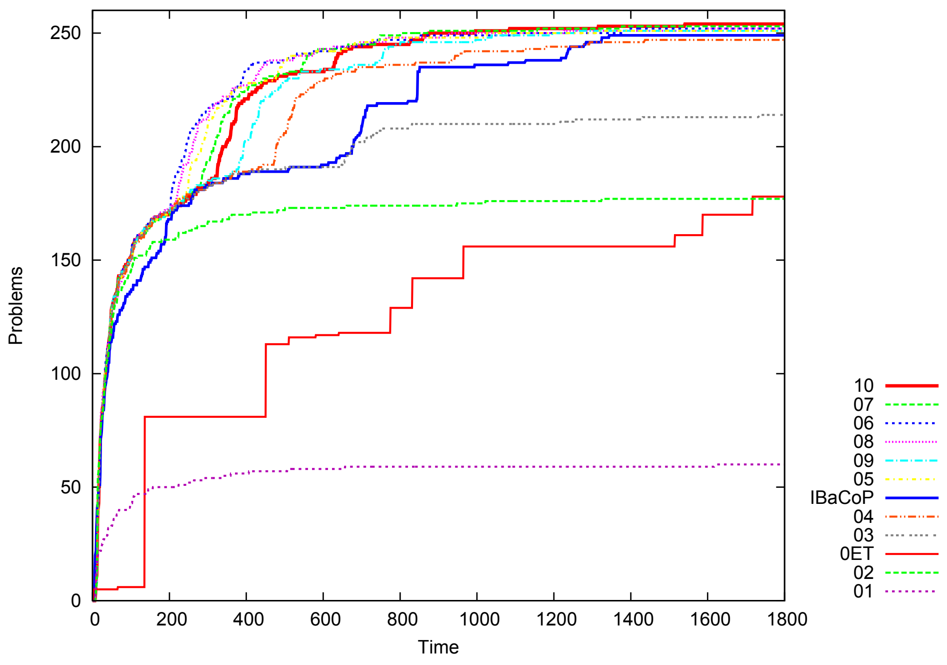


Figure 5.2: Number of solved problems by different Portfolio configurations. OET means the set of 28 planners; IBaCoP means the selection after applying QT-Pareto Score filtering strategy; and 01 to 10 references to the parameter N of a portfolio created with the *Best N Confidence*

highlight for $N = 10$, that the benefits arise only at the end of the time and that it even gets a negative gain when the running time is around 300 seconds.

Summarizing, if you consider more than four planners, a dynamic portfolio at least achieves the same results as a static strategy. These results show that the best strategies are for $N = 7$ and $N = 8$, considering that the test set is limited only for a IPC-2011.

Figure 5.4 shows the number of problem solved in the end with the different strategies (1,800 seconds in Figure 5.3). The red line is the number of solved problems with the initial 28 planners (OET strategy) and the green line is the number of problems solved with the QT-Pareto Score filtering (QT). The strategies from 5 to 10 are close to them. The portfolio with 5 planners solves fewer problems, but achieves better quality solutions as previously seen in Table 5.1. The portfolios with more than 5 planners also solve more problems and obtain a better quality score with respect to IBaCoP.

The results of this experiment show a classification method at least is as good as the static strategy when the number of planners is large enough (in this case, 5). The classification model guarantees accuracy in the selected planners and only wastes some time in computing the features that are not extracted from the pre-processing of FD. Henceforth, we consider a value of $N = 5$ for the following experiments with the *Best N Confidence* strategy.

Table 5.1: Results of the *Best N Confidence* Portfolio Configuration strategy selecting among one to ten planners (N parameter); in bold, the best results per domain and in blue the best strategy in more domains.

domain	10	09	08	07	06	05	04	03	02	01
barman	19.76	19.78	19.77	19.72	19.65	19.65	19.87	**19.87**	19.21	17.87
scanalyzer	19.58	19.56	19.80	**19.93**	19.50	19.50	19.32	18.64	17.85	5.44
parcprinter	19.99	19.98	19.99	**20.00**	19.99	19.94	19.84	19.23	18.36	0.00
pegsol	**20.00**	19.95	19.95	19.64	19.64	19.64	19.72	19.77	18.77	0.00
wood.	19.79	19.79	19.79	**19.87**	19.79	19.79	19.75	17.76	16.71	1.78
openstacks	19.65	**19.72**	**19.72**	19.44	19.44	18.45	18.16	18.04	14.57	6.57
nomystery	18.82	18.84	18.89	**18.91**	18.88	17.96	17.83	16.79	15.47	6.93
parking	17.62	17.73	18.13	17.90	17.90	17.79	18.02	**18.27**	16.01	4.52
tidybot	17.72	**17.75**	16.73	17.18	17.15	17.62	17.46	16.07	3.00	1.00
sokoban	18.68	18.72	18.52	18.57	**18.76**	18.48	15.14	4.37	0.00	0.00
transport	12.99	13.70	13.54	15.29	15.84	**15.90**	15.59	14.46	7.92	2.82
elevators	19.52	**19.66**	17.04	15.80	14.68	13.77	13.15	6.70	2.82	0.00
visitall	**13.78**	13.61	13.73	13.53	13.36	13.16	10.25	5.86	2.90	1.75
floortile	**8.88**	7.88	7.88	6.87	6.83	5.81	5.83	5.90	5.90	4.90
Total	**246.77**	246.67	243.49	**242.64**	241.41	237.43	229.93	201.71	159.50	53.58
#	**254**	251	252	**253**	253	244	240	211	186	65

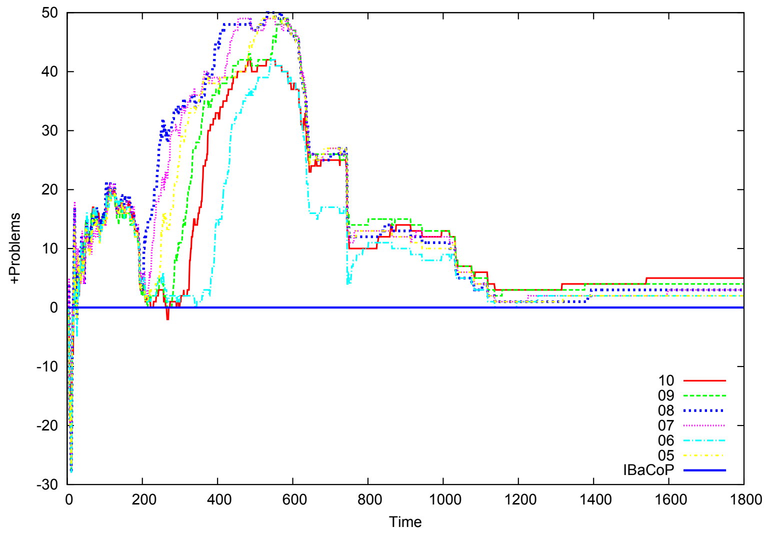


Figure 5.3: Problems' gain from five to ten planners selection against QT-Pareto Score filtering strategy at IPC-2011 (IBaCoP).

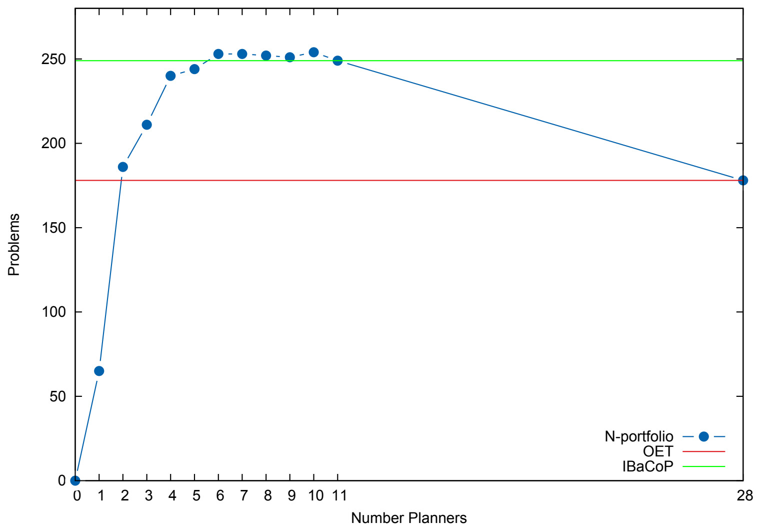


Figure 5.4: Number of Solved Problem per N-porfolio configuration. Red line is the number of solved problems per OET configuration (the final point of the blue line) and the green line (IBaCoP) is the number of solved problem per static configuration from the QT-Pareto Score filtering strategy (the point at 11 planners).

5.4 Portfolio Configuration Strategy Proposals

We have defined several strategies for the configuration of the portfolio in this thesis. The list of the strategies is ordered depending on the use they make of the knowledge provided by the predictive models. 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 multi-core track at 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 feature selection model, was the winner of the sequential satisficing track of IPC-2014 (For more details 6.6).

IBaCoP-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:

Random 5 Planners (Rand): This strategy is one of the baselines for comparing 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, 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 best 5 planners in terms of score. (i.e., LAMA-2011, PROBE, FD-AUTOTUNE-1, LAMA-2008 and FD-AUTOTUNE-2). Then, the time is assigned equitably. We want to see if using the best 5 planners is better than making a per-instance selection of 5 planners.

5.5 An Algorithm for Portfolio Construction

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 during the construction phase. Algorithm 2 shows how to use these strategies (*Equal Time (ET), Best N Confidence* (BN) and *Best N Estimated Time*) to configure the portfolio for a given planning task.

Algorithm 2: 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 uses 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 name of the planner. 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.

5.6 Summary

This chapter presents a selection and evaluation of different configuration strategies (configuration target) in a portfolio. The state of the art has two different branches: static and dynamic. Static configurations require a set of base planners, their times and their relative order, and these decisions are taken before the portfolio is built. Nonetheless, each criteria could be decided from among several strategies, for example, uniform time distribution and random order with all base planners. The drawback of the selection of this type of configuration target is that it could be far away for the VBS. The best approximation of achieve VBS is a dynamic configuration per problem, in which case the planners, their times and their orders have been decided at the time of execution. The closest approximation arises in SAT, where the portfolios always select the best solver per task. This was an unseen approach in AP until IPC-2014 when our approximation and AllPaca (Malitsky et al., 2014) appeared. However, the AllPaca approximation keeps the idea of selecting one planner. In contrast, we propose several strategies to select more than one planner. This novel idea requires the number of selected planner to be known. We determine the number of planners in an experimental evaluation to fix a set of two strategies (one strategy with the classification and another one including the regression task). Additionally, we establish a general algorithm to merge a uninformed strategy and the two proposed ones, in accordance with the requirements.

6

Experimental Evaluation of Instance-Based Configured Portfolios

Chapter 3 showed that, when creating portfolios, a main issue consists of defining the initial set of base planners from the whole set of available ones. In this chapter, the *QT-Pareto Score Filtering* was established as a method of selecting this initial set, maintaining a high level of diversity in the set of planners selected. Then, in Chapter 4 we demonstrated that accurate empirical performance models can be created to predict the behavior of the planners over a given task if we characterize the input tasks correctly. Finally, in Chapter 5 we described how these empirical models can be used in different configuration strategies to create portfolios.

In this chapter, we summarize the experimental results of these configuration strategies, creating portfolios based on the conclusions obtained in the previous three chapters, comparing the overall performance of planner selection, predictive models and portfolio strategies.

Firstly, we present an overview of the proposed ideas on the creation of planning portfolios, the experimental settings and the benchmark suit, following, the results of all strategies. Additionally, in Section 6.4 and 6.5, we provide an analysis of the diversity of the planner selection achieved by some configurations. We conclude the chapter with a brief summary of the IPC-14, in which our planners were prominent participants.

6.1 Scope of the Evaluation of Different Planning Portfolios

An ultimate goal of this thesis is the creation of a per instance-based configured portfolio in the general scope of a portfolio design seen at Section 2.6. To achieve this goal, we have evaluated the different portfolio strategies described in Section 5.4, which permits different portfolio configurations to be created. IBaCoP2 and IBaCoP-B5E were run with two predictive model versions, one trained with all of the features (f-all) and the other one trained with the selected features (f-14) (See Figure 6.1 as a summary of the developed strategies). We present other strategies to compare the effectiveness of the predictive models. The Random strategy was run 5 times and the average is reported. In addition, we have included the Jasper and Mercury planners in the comparison. These planners also raised in a sequential satisficing track in IPC-2014. Mercury (Domshlak et al., 2015) was the second best planner in terms of IPC score and Jasper (Xie et al., 2014b) was the second best planner in terms of problems solved (coverage). We summarize their main characteristics bellow:

- Mercury (Domshlak et al., 2015) is a sequential satisficing planner that is based mainly on the red-black planning heuristic. Red-black planning is a systematic approach to partially delete relaxation, taking into account some of the delete effects: red variables take the relaxed (value-accumulating) semantics, while black variables take the regular semantics. This planner, additionally, has a version for the agile competition.

- Jasper (Xie et al., 2014b) is a satisficing planner built on LAMA-2011 that includes two main additional capabilities. First, it implements an improved search algorithm, called Type Exploration based Greedy Best-First Search with Local Search (Type-GBFS-LS) (Xie et al., 2014a). This technique always expands a node which is the closest to a goal state according to a heuristic h. Second, it uses Diverse Anytime Search (Nakhost and Müller, 2010, Xie et al., 2013) which is a meta algorithm designed for solution improvement. This planner also has a version for the agile track.

6.2 Settings

As the test set we have used all the benchmarks of the sequential satisficing track in IPC-2014. This test set comprises 14 domains with 20 problems. These domains can be classified into several categories:

1. Previous IPCs: *barman*, *floortile*, *openstacks*, *thoughtful tidybot*, *transport* and *visitall*. These domains have been used in the training phase of this thesis, but the problems are totally new.

2. New domains: *ChildSnack*, *GED*, *Hiking* and *Tetris*. These domains are not used in the training phase.

3. New domains with conditional effects: *Cave Diving*, *City Car* and *Maintenance*. These domains present a difficulty for the portfolio construction because the initial set of planners does not support conditional effects.

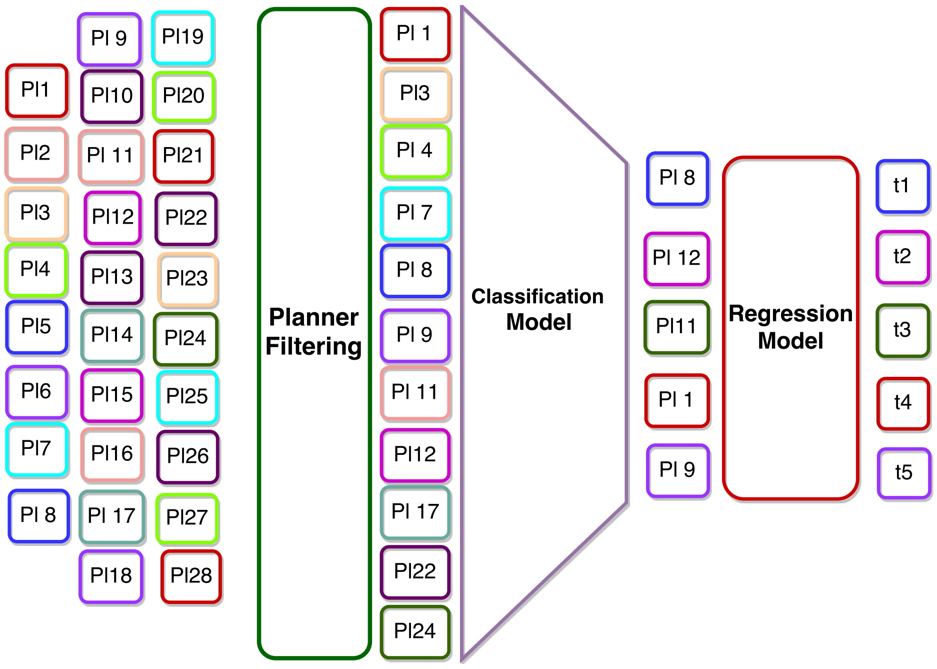


Figure 6.1: Experimental Strategies

To handle domains with conditional effects, we have included a parser that translates tasks with conditional effects into an equivalent planning task without this property. This translator is based on a previous translator ADL2STRIPS (Hoffmann et al., 2006). Specifically, we have implemented the compilation that creates artificial actions to effect evaluations (Nebel, 2000). Furthermore, many of the 11 candidate planners were built on the FD framework, which among other things, separates 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 process 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.

The experiments follow the settings of the IPC-2014 in sequential satisficing track. This track covers classical STRIPS planning (non-durative actions) with actions having associated non-negative costs (not necessarily uniform), negative preconditions and conditional effects. Each planner is given 30 minutes to solve each problem. The problems should be solved within reasonable time. The goal of this track is to find low-cost plans, in which the cost is defined as the sum of the costs of each plan's actions. The memory limit is 4GB of RAM.

IBaCoP configurations require a feature extraction phase. This process was limited to 4 GB of RAM (following IPC competition rules) and 300 seconds (which is approximately the maximum time used in the training set to obtain the features, as described in Table 4.12). The time needed to extract the features is included in the execution of the portfolio where, in the worst case, the feature extraction process takes 300 seconds and, therefore, the candidate planners have only 1,500 seconds to run. If the system does not extract the features in the 300 seconds, the input features are treated as missing values.

6.3 Results

Table 6.1 shows the results of all evaluated planners using the IPC quality score. The overall best planner is IBaCoP2 (f-all), followed closely by IBaCoP and IBaCoP-B5E (f-all). The difference between these configurations is negligible. All of the configurations using predictive models are much better than Default or Random baselines. 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 21 or more points higher in all cases.

Table 6.1: Results in terms of quality score. This table shows the second best planners: Mercury, the second planner in terms of quality and Jasper, the second planner in terms coverage in the IPC-2014. Following columns present a portfolio with the best 5 planners (Def.) and a random portfolio with 5 planners (Rand.) The remaining columns are our approximations, IBaCoP is the results of the QT Pareto Score Filtering, IBaCoP2 is our approximation with the classification model and IBaCoP2-B5S is our approximation with the classification and regression models.

						IBaCoP2		**IBaCoP2-B5S**	
	Mercury	**Jasper**	**Def**	**Rand**	**IBaCoP**	**f-all**	**f-14**	**f-all**	**f-14**
Hiking	18.96	18.17	18.78	18.07	19.25	**19.63**	18.51	19.63	18.39
Openstacks	**19.64**	18.76	19.25	17.23	17.35	17.38	17.74	17.37	17.74
Thoughtful	12.73	16.37	19.15	17.60	**19.17**	18.15	17.41	18.23	18.37
GED	**19.46**	17.95	16.40	14.22	17.31	17.70	17.70	17.70	17.70
Parking	18.14	17.22	**18.18**	12.47	17.89	18.16	17.20	18.17	17.23
Barman	14.61	**18.97**	17.17	14.10	16.79	16.85	15.44	16.87	15.44
Maintenance	5.72	10.79	12.52	15.27	**16.45**	16.21	16.36	16.25	15.56
Tetris	**16.37**	16.14	9.37	11.49	13.60	15.69	15.09	13.55	14.13
Childsnack	0.00	0.00	2.67	10.16	**19.50**	19.23	19.27	19.36	19.23
CityCar	4.10	11.03	4.96	9.77	11.43	**14.36**	12.37	12.57	10.73
Visitall	**20.00**	15.36	13.68	12.72	15.24	9.94	8.00	8.01	8.01
Transport	**19.87**	12.02	6.90	8.51	10.25	11.53	11.56	11.13	12.14
CaveDiving	7.00	**8.00**	7.00	7.00	6.30	7.00	7.00	7.00	7.00
Floortile	2.00	2.00	4.14	9.39	16.22	15.28	17.24	**17.46**	12.04
total	178.59	182.78	170.16	177.99	216.75	**217.11**	210.88	213.31	203.70

However, the QT-Pareto Filtering mechanism maintains the diversity, obtaining very good results. But again, increasing diversity will improve the results: the IBaCoP system, which uses the 11 planners, is better than the Random and Default configurations that use 5, although the latter have more running time per planner.

Therefore, reducing the number of planners is feasible if this reduction is informed by accurate predictors of the planner's behavior. For instance, the selection based on the predictive models, IBaCoP2 is able to perform better than IBaCoP, as it does not lose much diversity (IBaCoP only solves 3 problems more) while increasing the quality of the plans (since the planners in IBaCoP2 have more time). Once the set of 5 planners has been selected, the regression models do not contribute to obtaining a better performance. The task of estimating the run time needed to solve a problem is more difficult than the classification task (Schwefel et al., 2013). Additionally, given that the aggregated time predictions could exceed the time limit, our proposal re-scales 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 on them. These domains are Cave Diving, Child-Snack, CityCar, GED, Hiking, Maintenance and Tetris. The results show that the IBaCoP2 and IBaCoP-B5E generalize very well in unseen domains. In two domains they solve all the problems (GED and Childsnack), and in the other domains they solve the same or a fairly close number of problems as other planners.

The comparison between the results of the Def approach (the best 5 planners in terms of performance) with IBaCoP2 also shows the importance of the selection of different and diverse planners. For instance, the results in Table 6.1 show more than 44 points in terms of quality between both approaches. In addition, the comparison with Rand (the random selection of 5 planners) demonstrates that the predictive models select the planners properly to run in the available time, obtaining 30 points more than the random selection. Interestingly, the random selection performs better than selecting the best planners by score, because randomness provides diversity. The results in coverage enforce the previous ideas: Def solves 193 problems and Rand is slightly better (207 solved) while IBaCoP and IBaCoP2 solve more than 245.

Figure 6.2 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 this figure, the planners show two different behaviors. On the one hand, an asymptotic growth 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. That is the extreme case of Jasper, 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 IBaCoP-B5E, which selected a diverse set of planners, show a growing behavior throughout the time.

Table 6.2 presents the number of problems solved for 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. The optimal selection of 5 planners for each planning task would lead to 253 problems solved. IBaCoP2 is close to this optimum,

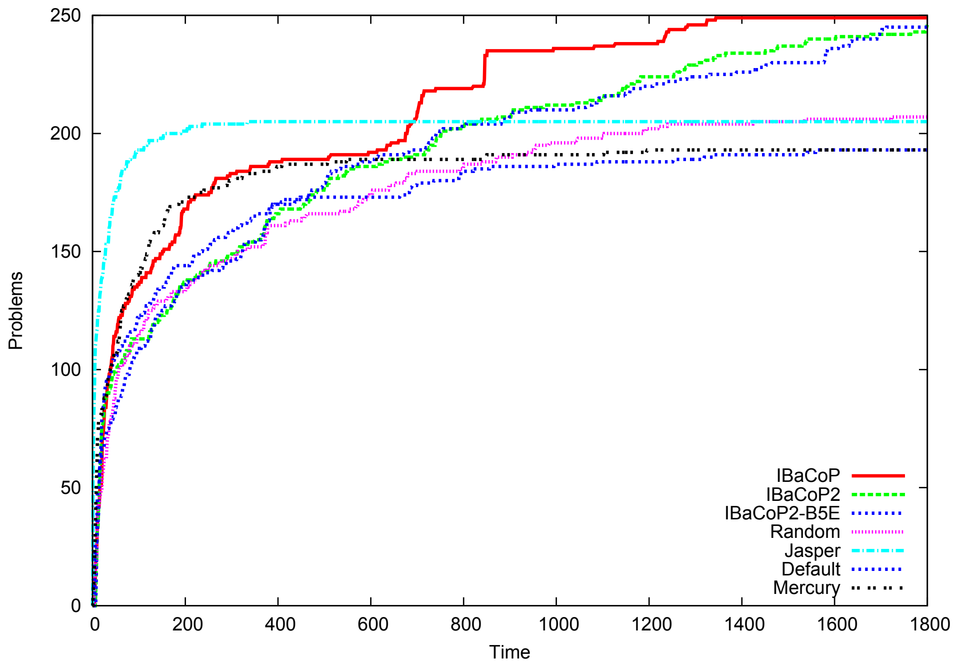


Figure 6.2: Comparison of IBaCoP configurations (IBaCoP, IBaCoP2 and IBaCoP2-B5E), the baseline configurations (Random and Default), and the Mercury and Jasper planners.

confirming its ability ti select 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.

6.4 Selection of Planners: per Domains vs. per Problem

In the previous section, we demonstrated that the diversity of the planners is the main element when deciding a portfolio configuration. A simple reason is that some planners may be good solvers in some domains, but not so good in others. However, in this section, we want to demonstrate that the planners might not be classified by how good they are at solving specific domains, but at solving 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 the larger size of the problems. Nevertheless, this increase affects the learning features at a different scale and intensity.

Therefore, in this section we discuss the diversity of the selected planners by IBaCoP2: we want to demonstrate empirically that, in the same domain, the classification model selects different subsets of planners for different problems, highlighting the importance of configuring the portfolio for each problem in each domain, instead of only for each domain (Gerevini et al., 2014), or fully static (Helmert et al., 2011). The importance of configuring a portfolio per problem is that the set of planners selected

Table 6.2: Results of the candidate planners defined in Table 3.3 and the maximum number of problems that can be solved by the complete set of these planners (VBS).

	lama11	probe	FDA1	lama08	FDA2	lamar	arvand	fdss2	ya2-mt	LPG	M	VBS
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**

can be better adjusted to the problem, using fewer planners, and providing more execution time to each planner. This contrasts to a per domain configuration, in which more planners must be selected to ensure that they cover the whole range of different problems in the domain, so less time must be assigned to each planner.

Figure 6.3 shows the diversity of the planners according to the selection made by IBaCoP2 (blue dots for f-14 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 in solving problems, but it is not always 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 6.3 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 states 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).

6.5 Confidence in Planner Prediction

In addition to the previous analysis, we wanted to carry out the underlying mechanism to achieve the diversity of planners. Remember that planners are selected based on the confidence of the success prediction. Therefore, in order to achieve different 5-planner

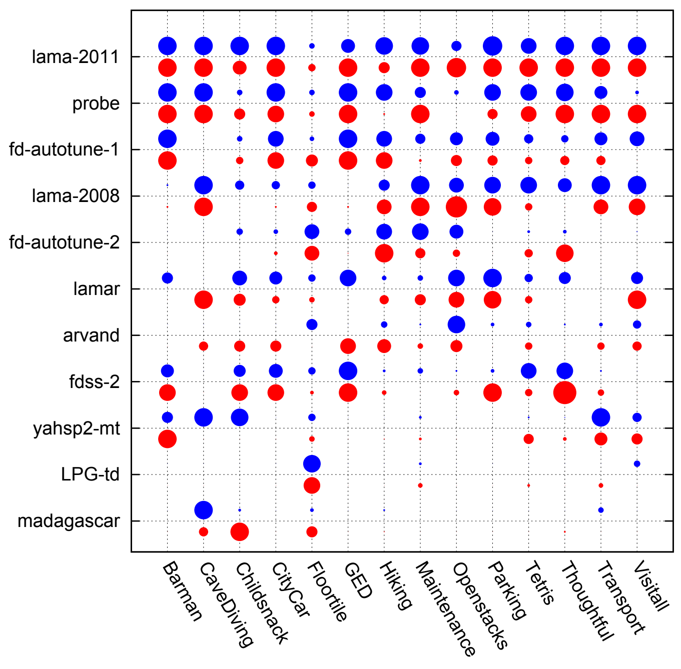


Figure 6.3: Proportion of the number of times each planner has been selected in a domain. In red dots, the proportion for IBaCoP2 (f-all), and in blue dots, the proportion for IBaCoP2 (f-14).

sets in the same domain, the ranking of the confidence of the prediction 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 6.3. This domain is a simplified version of the well-known Tetris game.

A heatmap with the confidence of the prediction of success appears in Figure 6.4. At a glance we realized that in general, a planner with a higher success rate in training time obtains greater confidence, but confidence ranking varies throughout different problems in the same domain. Another way of reading the picture is that the 5 darkest squares per column make up 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.

Table 6.3: Number of times a candidate planner has been selected by the two different classification models (f-14 and f-all)

	f-14	f-all	Average	±	SD
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

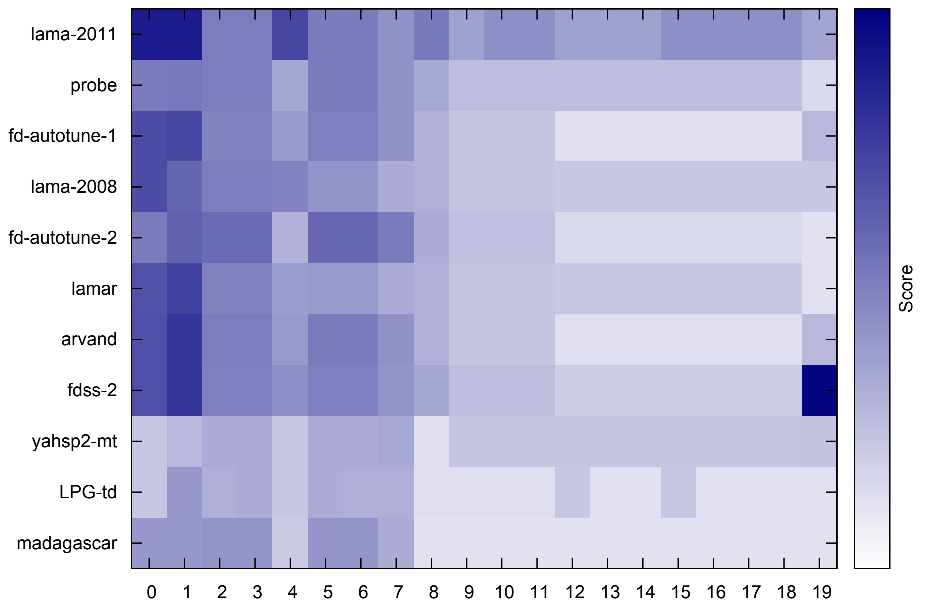


Figure 6.4: Success prediction confidence provided by the classification model (f-all) for each planner and problem in the *Tetris* domain. Scale goes from 0.0 (white) or no confidence at all to 1.0 (dark blue) or complete confidence

6.6 The 2014 International Planning Competition

In this section, we describe the planners derived from this thesis and that competed in IPC-2014. The submitted portfolio configurations are not exactly the same as those reported and evaluated in this dissertation, since the set of base planners and features used to describe the planning tasks have been amended since the competition. We have set out the full results of IPC-2014 in Appendix D. The IPC results are adequate for the evaluation of techniques, although the results may have been biased due to bugs and execution errors present in many planners and domains when they were submitted. All of these errors were fixed in the experiments reported in previous chapters.

1. Sequential Satisficing track

    - IBaCoP (Cenamor et al., 2014a): Static portfolio with 12 planners. In the competition, it included RANDWARD (Olsen and Bryce, 2011) and FDSS-1, but did not select LAMAR.
    - IBaCoP2: Dynamic portfolio. The planning tasks were characterized with fewer features, and the best predictive model for this set was Random Forest (Breiman, 2001). The same base planner selection as IBaCoP was submitted.

2. Sequential Sequential Multi-core track

    - IBaCoP: The same portfolio configuration as for the Sequential Satisficing track was used, but each of the 4 threads defined in the track run 3 planners with 1 Gb.
    - IBaCoP2: The same planner as in the Sequential Satisficing track, but three threads run 1 planner; and one, 2 planners.

3. Sequential Agile track

    - IBaCoP: This portfolio uses the 12 planners as the previous IBaCoP. However, we assigned the average time to find the first solution in 300 seconds as running time. The order of execution for the planners is given by this average from less time to larger values. Even though all planners are included, in practice, only a few of them will have the chance to run, until consuming the time limit of 300 seconds.
    - IBaCoP2: This portfolio uses the learned model to select 5 planners; the order of the planners is decided from the confidence and the time for each planner is the same for the five planners. The training set only includes the results of the planners up to 300 seconds.

4. Learning Track

- LIBaCoP Cenamor et al. (2014b): is a portfolio configurable with a predictive classification model similar to IBaCoP2. However, this predictive model is built *Ad hoc* per domain (using only problems of this domain for training purposes). The 5 planners with higher confidence are selected, and they are ordered following this confidence. Then, the running time is divided uniformly among them. The candidate planners are the same to IBaCoP and including additionally LAMAR, DAE-YAHSP (Dréo et al., 2011) and SGPLAN (Bonisoli et al., 2015) (15 planners).
- LIBaCoP2: is a portfolio that includes a regression model to predict the running time for the planners selected by the classification model. This strategy is similar as IBaCoP2-B5S but the predictive models are built only for one domain and the candidate planners are the same as in LIBaCoP.

In the competition, all the dynamic configurations presented a control memory bug, so the available memory was exceeded. This issue has been solved in all of the experiments reported in this thesis, improving the results.

In this chapter we have highlighted the performance of the planners we developed in this thesis. We made an empirical comparison of all the approaches we considered and highlighted our results against the most promising planners in IPC-2014.

6.7 Summary

The empirical results of the application of the contents of the three previous chapters to build a dynamic portfolios. We summarize the strategies, and we explain the setting and the set of planners that we compare with. This reported evaluation shows remarkable results: our approximations achieve 20 more points in quality than the second planner in the IPC-2014. Despite the differences, these results confirm that our approximations are the state of the art in satisficing planning. Reducing the number of planners is feasible if this reduction is informed by accurate predictors of the planner's behavior. The generated predictive models could be generalized across unseen domains as we reported in the results. Additionally, we present an analysis of the results to discover the selected planners with the learned models per domain (f-14 and f-all models). Furthermore, we report the evaluation of confidence in a domain to understand how the predictive model is working. In the last part of the chapter, we explain the submitted version of the portfolio to the IPC-2014, and the small differences from the reported strategies in this dissertation.

7

Performance Modeling of Planners in Homogeneous Problem Sets

One of the main objectives of this thesis is to develop a per-instance configured portfolio. Therefore, we have to show that empirical performance models are able to recognize different planner performances even when planning tasks are fairly similar in their structure. However, the evaluation shown in previous sections is not enough to demonstrate this ability. IPC benchmarks are not suitable for this propose because, given a domain, the problem set is usually created in such a way that they tend to increase the difficulty of the problem. For this reason, in this chapter we propose to train performance models with homogeneous problem sets in order to verify whether some of our features are able to characterize planning tasks even when they have the same syntactic structure. The issue with these kinds of sets is that planners might have significant differences in performance for some problems, however, problems could produce the same value for superfluous features, such as the number of goals, objects, actions, etc. Consequently, instances characterized only by these features are indistinguishable. In this chapter, we demonstrate that this situation does not happen if we consider our whole set of features (de la Rosa et al., 2017).

In addition, this study takes into account that a problem may not have the same difficulty for each planner. For this reason, models will be created for individual planners. The selected planners for this study are LAMA-2011, PROBE and MERCURY. These planners obtained good results in the competitions in which they have participated. LAMA-2011 was the winner of the IPC-2011, PROBE was the second best planner in terms of coverage in IPC-2011, and MERCURY was the best individual planner and achieved the Runner-Up and innovate planner award in the last competition (IPC-2014).

This chapter is organized as follows. First, we present an experimental preview in which we identify the homogeneous test sets that are of interest for this study. After that, we explain our proposal, a experimental procedure to evaluate these interesting problem sets. Then, we apply the procedure in six of these problem sets and show a summary of the relevant features in each one.

7.1 Experimental Preview

Firstly, we propose a proof-of-concept to demonstrate that homogeneous problems may require different run times to be solved by the same planner. Therefore, a first step is to find a group of domains that satisfies this situation. To test it, a problem generator should be available for the domains, so problems can be easily generated. As an additional condition, the same input configuration for these problem generators should generate different planning problems. These requirements restrict the domains we can use.

For the evaluation, we have initially considered the domains from IPC-2011 and IPC-2014 for which we have found random problem generators. The first step we have made towards the selection of interesting problem sets is the generation of a set of 30 random problems with the same input parameters for each domain. The size of the problem, determined by these input parameters, was manually adjusted to ensure a non-trivial problem that, in most cases, can be solved within $1,800$ seconds time limit. Planners were run on these problem sets to visualize the performance profile of each domain.

Table 7.1 shows the coefficient of variation (i.e., the ratio of the standard deviation to the mean, $C_v = \sigma/\mu$) for the 13 domains that fulfill the requirements for the problem generator. With this relative measure of dispersion one can compare performance in different domains and realize which of them have more variety in their performance profile. A large coefficient means the runtimes are fairly different among the problems; opposite, values close to zero means that all problems are solved in almost the same running time. We have marked the top 6 C_v in bold. We have focused on these planner/domain performances for the rest of the evaluation. The selected problem sets are from *Barman*, *Depots* and *Floortile.*

The results of this table show that *blocksworld, transport* and *rovers* domains have small values for the coefficient of variation for all of the planners, which means that the "complexity" of the problems of these domains is quite similar for all planners. However, different planners may have totally different behaviors in the same domain. For instance, in domains such as *tpp* and *depots*, MERCURY and LAMA-2011 planners have large values of C_v while PROBE has small values. The planners based on the same system (i.e., LAMA-2011 and MERCURY) could have a similar performance and consequently, similar values of C_v. The inverse situation appears in the *floortile* domain in which the PROBE planner has high values of variation and the other planners low values. In summary, our hypothesis is that planning tasks of the same size might not need the same time to be solved, and this time depends on the planner, the domain and the capability of the problem generator to produce problems of the same size but a significantly different search space.

Other aspects are the number of problems solved and the average time to carry out the planning tasks for all planners (Table 7.2). Most of the problems of each domain are solved for at least one planner except for the *nomystery* domain. This domain only

Table 7.1: Coefficient of variation ($C_v = \sigma/\mu$) of the time with 30 problem per each domain with the same size. In this table appears in bold the 6 bigger values.

No.	Domain	Mercury	Probe	Lama
1	*barman*	**2.86**	**2.02**	**2.63**
2	*blocksworld*	0.54	0.14	0.76
3	*depots*	**2.97**	0.08	**2.00**
4	*elevators*	0.05	0.22	1.03
5	*floortile*	0.82	**1.88**	0.41
6	*parking*	0.20	0.69	0.26
7	*transport*	0.02	0.00	0.01
8	*spanner*	0.04	0.21	0.08
9	*rovers*	0.49	0.17	0.42
10	*gripper*	0.72	0.04	0.65
11	*tpp*	1.57	0.17	1.07
12	*satellite*	1.25	0.39	0.30
13	*nomystery*	0.25	–	0.41

gets solutions from Mercury and Lama-2011 planners, but these planners do not solve all the problems: the maximum number of problems solved is 14. The execution time of the problems depends on the planner, the domain and the size of the problem in general. For example in the *transport* domain, Mercury has an average time of 20.30, very far from the values obtained by the others (Probe time is close to 690 seconds and Lama-2011 time close to 200 seconds).

These results provide insights into the differences between problems of the same size. Additionally, they show that the planners do not take the same amount of time to solve them so it is hard to predict this running time. This fact reinforces the idea of a proper characterization of the planning task to truly predict the performance of the planners per problem, and not only per domain. In the next section, we explain the experimental procedure and the evaluation made in the selected domain/planner combinations.

7.2 Experimental Procedure

We propose an evaluation strategy following Procedure 7.1. The first step is to generate 170 additional problems of the same size and structure (the problem sets are those from the highest C_v values in Table 7.1). After that, we need to separate the problems into two parts, the first part is the "easy" one; this group has to ensure that the planning task is solved by the planner and the time needed to solve it is under a fixed cut-off point. The second group is the subset of "hard" problems, including the unsolved ones. These problems need more time to be solved than a particular cut-off point. For the separation we have used Algorithm 3.

Algorithm 3 receives a list of problems (Π), a domain (d), a base planner (p), a list of times for the first solution per each problem (T) and a cut-off (c). The first

Table 7.2: Solved problems and average time to solve 30 problems in each domain for MERCURY, PROBE and LAMA-2011 planners. In bold the selected domains for the experiments and in blue the five concrete set of planner-domain with high values in terms of coefficient of variation.

No.	Domain	Mercury		Probe		Lama	
1	***barman***	30	36.97	26	65.92	30	76.60
2	*blocksworld*	27	97.30	30	154.03	26	122.00
3	***depots***	27	114.63	29	5.24	26	162.85
4	*elevators*	30	10.83	30	327.43	28	199.79
5	***floortile***	29	372.69	12	217.92	22	375.45
6	*parking*	30	28.77	20	786.90	30	52.73
7	*transport*	30	20.30	30	690.80	30	197.27
8	*spanner*	30	71.27	30	129.37	30	80.27
9	*rovers*	26	477.15	29	599.48	30	428.53
10	*gripper*	17	670.12	30	49.63	30	195.13
11	*tpp*	30	118.30	30	268.90	30	111.80
12	*satellite*	30	49.33	25	537.48	21	379.57
13	*nomystery*	14	3.86	0	–	4	40.00

Algorithm 3: Generating different datasets with homogeneous problems set.

```
Data: Problems ($\Pi$), Domain ($d$), Times ($T$), Planner ($Pl$), Cut-off ($c$)
Result: Problems list with label solve/unsolved $L = [\langle \pi_1, l_1 \rangle, ..., \langle \pi_c, l_c \rangle]$
ExecutionTimes = sortedList($\Pi$, $T$);
if $c <$ UnsolvedProblems then
    return LabelClass(ExecutionTimes)
else
    LabelList = SelectPositiveClass(ExecutionTimes,$c$);
    return LabelList
```

step consists of sorting the problems by the execution time in ascending order. If the unsolved problems have more than a c percentage, we always preserve the highest value. This procedure (*LabelClass*) labels the problem with $True$ when the time is $t \geq 0$; if, $t = -1$, the problem is labeled as $False$. If the problems solved are less than c, several problems are labeled as $False$ although the times are greater than zero. This procedure, (*SelectPositiveClass*), labels as $False$ the problems where the planner needs more time to solve the planning task. For the experiments, we have built several datasets with different cutoffs ($c = 95, 90, 75, 66$). The general idea of this procedure is to generate datasets by changing the instances labeled "True" or "False" according to the cut-off, similar to filtering the instances with a time limit lower than the initial one.

After that, (**Step 5**), there are five files with different cut-offs. We evaluate them by taking into account accuracy and the area under the ROC curve (AUROC). AUROC is a curve that evaluates the true positive rate against the false positive rate at various threshold values. ROC curves have an attractive property: they are robust to changes in class distribution (the proportion of positive to negative instances). The AUROC

Procedure:

1. Generate 200 problems (D) with the same size P_p
2. Run this 200 problems, D with each planner $pl_i \in P$ a time limit T
3. Generate a dataset per each domain D and planner Pl_i
4. Apply Algorithm 3 to generate different datasets per experiment
5. Evaluate a set of predictive model with previous datasets (*ZeroR, J48, Naive Bayes, Random Forest & Rotation Forest*)
6. Select the most balanced dataset
7. Delete features with a standard deviation with values lower than 1.0
8. Apply Feature Filtering Criteria
9. Show ranking and feature selection

Figure 7.1: Experimental procedure to evaluate homogeneous problems set

has an important statistical property: the AUROC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance, which is a close to the Wilcoxon test of ranks (Hanley and McNeil, 1982). The results of the evaluation are generated using 10-fold cross-validation.

The selected algorithms are a small set of all the available ones; in this selection we include a base model, ZeroR, based on the majority class; a decision tree, J48 (Quinlan, 1993); a bayesian classifier, Naive Bayes (John and Langley, 1995); Rotation Forest (Rodriguez et al., 2006), which carries out an analysis on the main components to transform the features before training a base classifier; and Random Forest (Breiman, 2001) which constructs a forest of random trees. In theory, a random classifier has an AUROC of 0.5, no matter the class balance, given that true and false positive ratios are independent of the number of positive and negative instances. In the evaluation, if the computed AUROC for a classifier is larger than 0.5, we can claim that it behaves better than random.

The next step, (**Step 6** & **7**), is to select the most balanced dataset, which is almost always when the cut-off is 66%. The dataset is cleaned by deleting all of the features that have a standard deviation of less than one point. As a consequence, the features with less different values are deleted from the datasets for the feature evaluation.

The feature evaluation (**step 8**) (Karegowda et al., 2010) determines the relevant features of the dataset. The techniques used are: Gain Ratio (GR) and Correlation based Featured Selection (CFS).

Definition 23 (Gain Ratio)**.** It is a standard discrimination criterion used in decision trees. A decision tree is a simple structure where non-terminal nodes represent tests on one or more attributes and terminal nodes reflect decision outcomes. The information gain ratio is used to select the test attribute at each node of the decision tree. This ratio is defined in the equation 7.2

$$GainR(Class, Attribute) = \frac{(H(Class) - H(Class|Attribute))}{H(Attribute)} \tag{7.1}$$

The attribute with the highest gain ratio is selected as the splitting attribute respect to the class. The non-leaf node of the decision tree generated is considered as a relevant attribute. Formula 7.2 represents the gain ratio per attribute in respect to the class in which $H(class)$ is the entropy of the class; $H(Class|Attribute)$ is the conditional entropy or quantity mutual information; and $H(Attribute)$ is the entropy of the attribute. The results of the feature evaluation will show the Gain Ratio and a ranking of features generated from the relative order of these values. The ranking procedure is provided by the Weka tool.

The second feature selection method is the Correlation based Feature Selection (Hall, 2000).

Definition 24 (Correlation based Feature Selection (CFS))**.** CFS is a heuristic for evaluating the worth or merit of a subset of features. This heuristic takes into account the usefulness of the individual features for predicting the class label along with the level of intercorrelation between them. The relevance of the features grows with the correlation between features and classes, and decreases with a growing inter-correlation (Hall, 1999). The CFS is defined by the equation 7.2

$$r_{zr} = \frac{k\bar{r_{zi}}}{\sqrt{k + k(k-1)\bar{r_{ii}}}} \tag{7.2}$$

where r_{zr} is the correlation between the sum of the feature subsets and the class variable (it is the heuristic of the feature that allow the merit to appear in the final selection), k is the number of subset features, $\bar{r_{zi}}$ is the average of the correlations between the subset features and the class, and $\bar{r_{ii}}$ is the average inter-correlation between the subset features (feature-feature).

This metric is used with search strategies such as Hill-climbing and Best First (Kohavi and John, 1997) that find a good subset of features in a reasonable time. This search starts with an empty set of features and generates all possible single feature expansions. The subset with the highest evaluation is chosen and expanded in the same manner by adding single features. The search stops when the algorithm finds five consecutive subsets that do not improve the results. The feature ranking will show a relative order among them.

The last step (**Step 9**) is to show two rankings of the feature selection. These methods are included in the algorithm to select the features and their significance in the dataset.

7.3 Experimental Evaluation

The experimental evaluation is made in three different domains following the experimental procedure explained in Section 7.2. The experiments are the six highest values according to Table 7.1. We should highlight three different cases taking into account that their coefficients of variations are the highest. The first one is the same domain (*Barman*) with all planners. The second one is when two planners (Mercury and Lama-2011) have high values and the other one, Probe, has a small value. The last case is the inverse case when the PROBE planner has a high value and the other planners have less variation.

7.3.1 Barman Domain - Mercury Planner

This domain consists of a robot that handles drink dispensers, glasses and a shaker. The goal is to find a plan of the robot's actions that serves a desired set of drinks. In this domain, deletions of actions encode relevant knowledge given that robot's hands can only grasp one object at a time and given that glasses need to be empty and clean to be filled. The size of the problems are as follows: 1 shaker, 2 hands, 15 shots, 5 ingredients, 10 cocktails, 5 dispensers and 3 different levels. The goal is to prepare 14 shots.

First, we show the execution times of all of the problems included with MERCURY. The execution times follow approximately a Chi-squared distribution (Huzak, 2011) in which most of the problems are solved in similar periods of times and a small number of them take the whole of the time available. Figure 7.2 shows that the 200 are solved in 1,000 seconds. However, the majority of the problems are solved before 20 seconds. In addition, more than 100 problems are solved in 10 seconds.

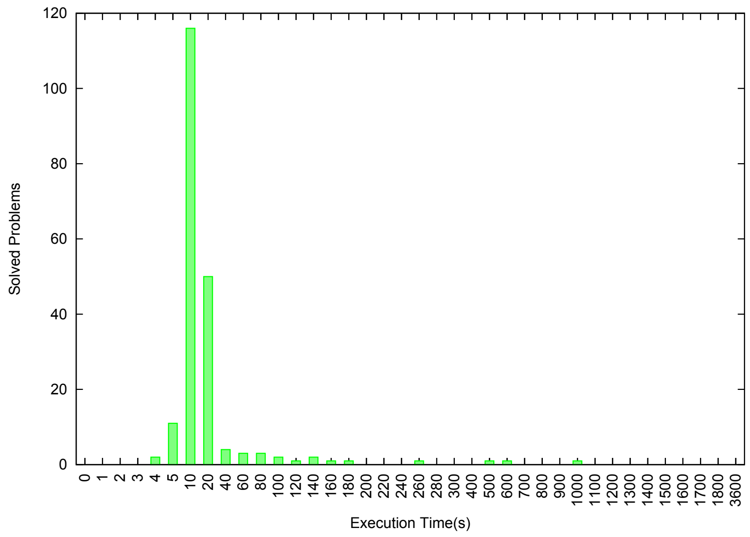
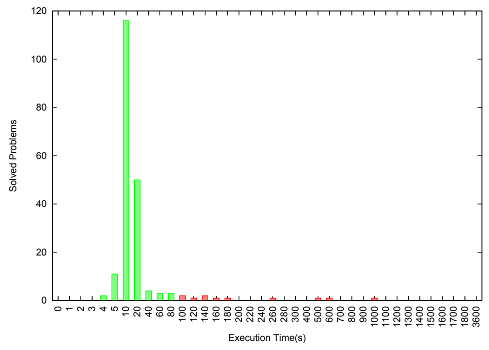


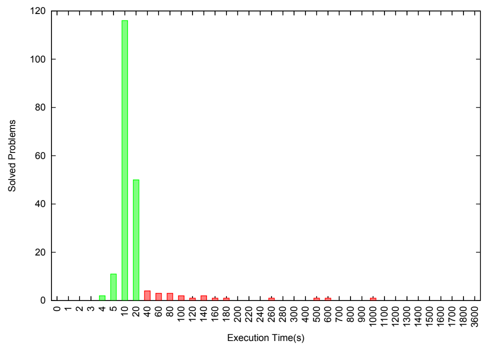
Figure 7.2: Execution time for the 200 problems of *Barman* domain with MERCURY planner.

Figure 7.3 shows the data in the four datasets with different cut-offs. In subfigure 7.3a, the problems with the first time greater than 100 are labeled as unsolved. In subfigure 7.3b, the problems with a time greater than 40 are labeled as unsolved in this case. The other two subfigures, 7.3c and 7.3d split the data into 20 seconds and 10 seconds respectively.

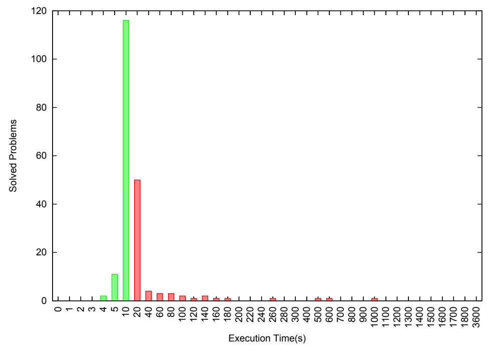
Table 7.3 shows the results in terms of accuracy and AUROC. The values of this table are always better than the base algorithm, ZeroR, increasing the value of AUROC up to around 0.7.



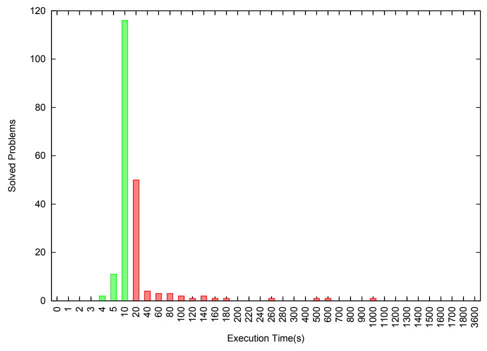
(a) $c = 95$



(b) $c = 90$



(c) $c = 75$



(d) $c = 66$

Figure 7.3: Execution time for the 200 problems of *barman* domain with Mercury planner when 95, 90, 75 and 66 are label as solved problem (True) in green.

The next step is to discover the knowledge implicit in these models and select a group of relevant features. The number of features that are included in this experiment is 56. A total of 33 based on the SAS$^+$ formulation, 6 from the fact balance, 7 heuristic values and 10 from the landmark graph. Several of these features do not have a standard deviation of more than one point. After deleting these features, the set decreases to 17 features. The following step is to discover the relevance of these features for the class with the two metrics proposed in the section 7.2 (Gain Ratio and Correlation based Feature Selection). These mechanisms work better when the proportions of the output class are balanced. In this case, we have selected the 66% dataset to carry out the analysis.

Table 7.4 shows a list of the features grouped by type (the first column of the table) and the two accomplished tests. CFS only selects 2 features and GR selects 8. It seems intuitive that PDDL or FD features are not informative enough for problem differentiation, in contrast to heuristic values or other types of features.

Table 7.3: *Barman* Results in terms of accuracy (Acc) and area under the ROC curve (AUROC) with the original data and a cut-off of 95, 90, 75 and 66.

	Original		95%		90%		75%		66%	
Algorithm	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC
ZeroR	100.0	-1.0	95.0	0.50	90.0	0.50	75.0	0.50	66.0	0.50
J48	100.0	-1.0	94.5	0.50	90.5	0.59	76.5	0.58	68.0	0.62
NaiveBayes	100.0	-1.0	77.0	**0.68**	71.0	**0.71**	65.5	**0.71**	67.0	**0.71**
RandomForest	100.0	-1.0	94.0	0.67	89.5	0.60	75.0	0.67	66.5	0.65
RotationForest	100.0	-1.0	95.0	0.51	90.0	0.61	79.0	0.60	70.0	0.64

Table 7.4: List of useful features in *Barman* domain with MERCURY planner, the ranking of the features, their gain ratio (GR) and the ranking provided by correlation based feature selection (CFS).

Type	Name	Ranking	GR	CFS
SAS – CG	Total edges	18	0.00	-
	Maximum output edge HV	11	0.00	-
	Maximum input weight HV	9	0.00	-
	Maximum output weight HV	13	0.00	-
SAS – DTG	Total edges	12	0.00	-
	Total weigth	15	0.00	-
	Maximum input weight	14	0.00	-
	Maximum output weight	16	0.00	-
FB	Balance ratio	4	0.11	1
	Balance distorsion	3	0.11	-
Heuristics	Additive	7	0.11	-
	Causal Graph	8	0.11	-
	Context-enhanced Additive	6	0.11	-
	FF	10	0.00	-
	Landmark-cut	17	0.00	-
	Hred-black	2	0.11	-
Landmarks	Number edges	1	0.12	2
	Maximum input edges	5	0.11	-

7.3.2 Barman Domain - Probe Planner

The next experiment is with the same domain, but changing the planner. The PROBE planner obtains the solutions about twice as fast as MERCURY. Figure 7.4 shows that the problems start to be solved after approximately 10 seconds, and several problems are not solved in $1,800$ seconds. Figure 7.4 details that the 70% of the problems are solved in 20 seconds and 80 problems more in the following 160 seconds (75% of the problems). The number of unsolved problems is 27 (13.5%).

Figure 7.5 shows the four datasets with different cut-offs. The negative class has 13.5% of the problems, and the original dataset is not modified until a cut-off of 75%

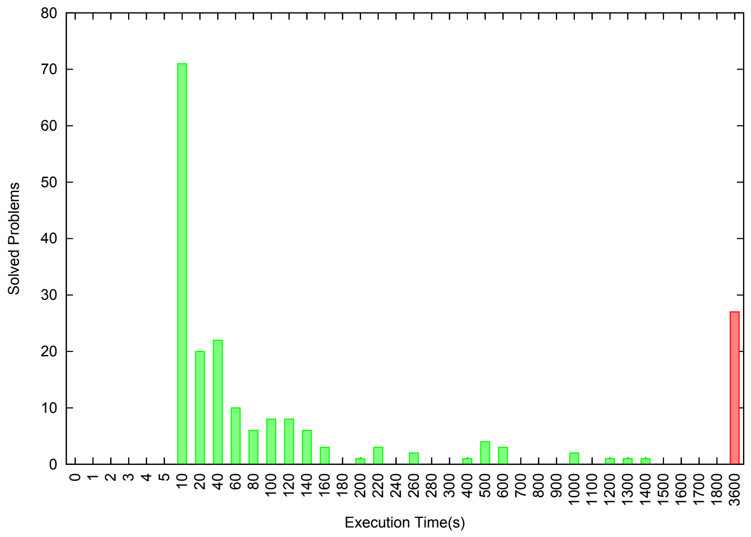
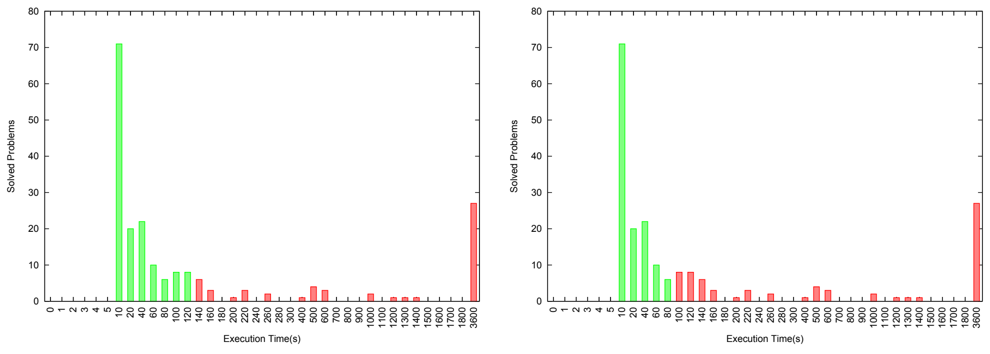


Figure 7.4: Execution time for the 200 problems of *Barman* domain with PROBE planner.

is applied. (a) Subfigure 7.5a splits the data into 140 seconds and (b) Subfigure 7.5b splits the data into 100 seconds.



(a) $c = 75$ **(b)** $c = 66$

Figure 7.5: Execution time for the 200 problems of *barman* domain with PROBE planner when 75% and 66% are labeled as solved problem (True) in green.

Table 7.5 shows the accuracy and AUROC using a 10-fold cross-validation. The best algorithm is Rotation Forest when the value of AUROC is 0.72 in the initial dataset and more than 0.60 in the other cases. These results improve the base models or a random classifier in all cases.

The next step is to discover the implicit knowledge in these models. The group of selected features are identical because the input attributes are the same. It only changes the output attribute that is created through the execution of the PROBE planner and the algorithm 3.

Table 7.5: Barman results in terms of accuracy (Acc) and area under the ROC curve (AUROC) with a with the original data and a cut-off of 75% and 66% with PROBE planner.

	Original		75%		66%	
Algorithm	Acc	AUROC	Acc	AUROC	Acc	AUROC
ZeroR	86.5	0.46	75.0	0.50	66.0	0.48
J48	85.0	0.58	72.5	0.48	61.5	0.59
NaiveBayes	69.5	0.61	65.0	**0.61**	55.0	0.54
RandomForest	85.5	0.70	69.5	0.58	63.0	0.57
RotationForest	85.5	**0.72**	**72.0**	0.61	63.0	**0.63**

Table 7.6 has different values in the two feature metrics. GR discards six additional features and two of them are the same as the MERCURY planner (maximum input/output weight at DTG graphs), however, the ranking value is totally different. The PROBE dataset takes the SAS$^+$ features into account before heuristic values on almost all occasions. The CFS method only considers 4 features as relevant, 2 more than the previous evaluation and 1 of them coincident (balance ratio). This feature is always selected with the two methods.

7.3.3 Barman Domain - Lama-2011 Planner

The next experiment is with the same domain and the LAMA-2011 planner. Figure 7.6 details the problems solved in 1,800 seconds. This planner solves close to 150 problems in less than 40 seconds and less than 50 problems from between 40 to 1,500 seconds. Five of the 200 are unsolved (2.5%).

Figure 7.7 represents the different proportions used. In Subfigure 7.7a, the problems that require more than 1,000 seconds to achieve the solution are labeled as unsolved. In Subfigure 7.7b, the problems with a execution time of more than 600 are labeled as unsolved in this case. The other two subfigures, 7.7c and 7.7d split the data into 20 or 16 seconds respectively.

Table 7.7 shows the accuracy and AUROC of these datasets, evaluated using a 10-fold cross-validation. There are two important things to highlight; the first one, the values of the AUROC are over the base algorithm in all cases; the second one is that for cut-offs equal to or less than 90%, there is a model that obtains an AUROC of around 0.70.

Table 7.8 shows the list of those features with a standard deviation of more than one. 4 of them do not contribute to the class, reducing the relevant feature to 12 in this case. As expected, none of the PDDL features appear in the list because they always have the same values. FD Instantiation features are not relevant enough for this domain.

The study of one domain with three different planners gives us several highlights. The best feature is the *Balance ratio*, very close to the other ones based on the fact balance. The heuristic values are close to the most important ones but they depend on

Table 7.6: List of useful features in Barman domain with PROBE planner, the ranking of the features, their gain ratio (GR) and the ranking provided by correlation based feature selection (CFS).

Type	Name	Ranking	GR	CFS
SAS – CG	Total edges	5	0.12	1
	Maximum output edge HV	3	0.15	2
	Maximum input weight HV	2	0.15	-
	Maximum output weight HV	4	0.15	-
SAS – DTG	Total edges	7	0.11	-
	Total weigth	6	0.11	-
	Maximum input weight	11	0.00	-
	Maximum output weight	10	0.00	-
FB	Balance ratio	9	0.10	3
	Balance distorsion	18	0.00	-
Heuristics	Additive	15	0.00	-
	Causal Graph	17	0.00	-
	Context-enhanced Additive	16	0.00	-
	FF	1	0.16	-
	Landmark-cut	8	0.10	4
	Hred-black	12	0.00	-
Landmarks	Number edges	14	0.12	-
	Maximum input edges	13	0.11	-

the planner. However, the only planner that considers the SAS$^+$ features as relevant is PROBE and it does not use this formulation.

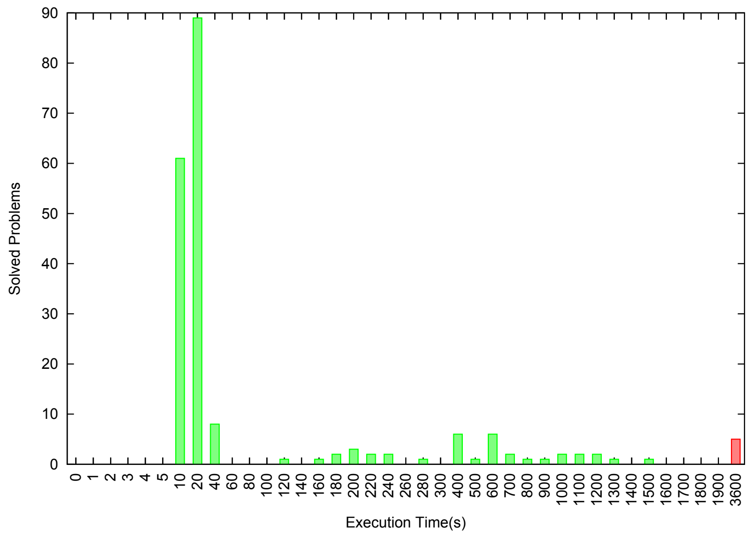


Figure 7.6: Execution time for the 200 problems of *Barman* domain with LAMA-2011 planner.

Table 7.7: *Barman* Results in terms of accuracy (Acc) and area under the ROC curve (AUROC) with a original data and a cut-off of 95, 90, 75 and 66.

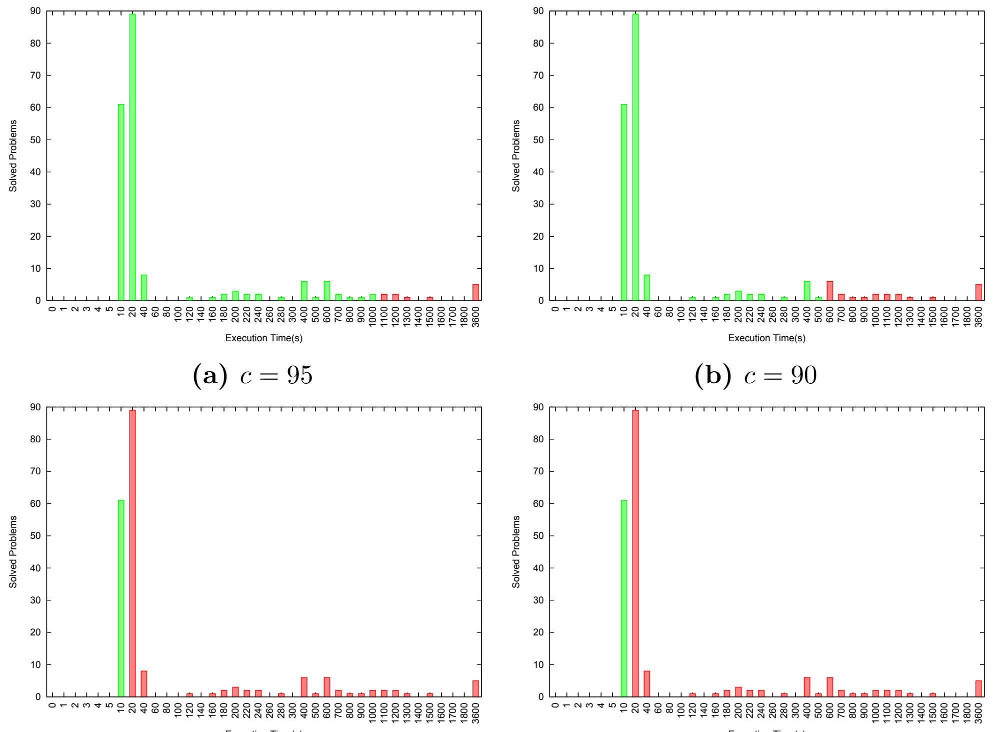
	Original		95%		90%		75%		66%	
Algorithm	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC	Acc	AUROC
ZeroR	97.5	0.24	95.0	0.50	90.0	0.50	75.0	0.50	66.0	0.48
J48	97.5	0.24	95.0	0.50	90.0	0.50	75.0	0.64	66.0	0.57
NaiveBayes	52.0	**0.46**	53.5	**0.58**	59.5	**0.70**	64.0	**0.70**	64.5	**0.71**
RandomForest	97.5	0.44	94.0	**0.58**	85.5	0.54	78.5	0.67	67.0	0.59
RotationForest	97.5	0.24	95.0	0.50	90.0	0.52	75.0	0.67	69.0	0.65

7.3.4 Depots Domain - Mercury Planner

This is an experiment in another domain, *depots*, with the MERCURY planner. This domain is a combination of a transportation domain and the *Blocksworld* domain. The transportation element of the task is to move crates from one depot to another using trucks. The *blocksworld* element arises due to the need to stack and unstack the crates, with the space on the 'table' being limited by the number of pallets at each location. Hoists serve as the function of the robot arm, including the mechanism by which crates are loaded/unloaded into/from trucks. The goal is to find a plan in which crates are stacked appropriately at their destinations.

The size of the problems are as follows: 2 distributors, 2 trucks, 6 pallets, 20 crates and 3 hoists and the goals are to locate all of the crates in the right place. Figure 7.8 shows the progression of the 200 problems with the MERCURY planner. This planner does not solve 26 problems (13%). Most of them are solved in less than 160 seconds.

Figure 7.9 represents the different proportions used. The percentage of unsolved



(a) $c = 95$ **(b)** $c = 90$

(c) $c = 75$ **(d)** $c = 66$

Figure 7.7: Execution time for the 200 problems of *Barman* domain when 95%, 90%, 75% and 66% are label as solved problem (True) in green.

problems is 13%, therefore 90% and 95% cutoff points do not make sense. Subfigure 7.9a(a) labels the problems as unsolved in times greater than 300 seconds and Subfigure 7.9b(b) has the split point at 120 seconds.

Table 7.9 shows the accuracy and AUROC of the different proportions of the training data. It is important to highlight that the 75% cutoff has an AUROC of 0.98, very close to the perfect value 1.0.

In this case the group of relevant features is different from the *Barman* domain. After the filtering of the standard deviation, 34 features are selected. This feature selection includes features from all types, one from PDDL, one from the FD instantiation, 14 from the SAS$^+$ formalism, 3 from Fact Balance, 9 heuristic values and 7 from landmarks. The feature from the PDDL group is the number of goals, however its contribution is not significant. The number of goals in *Depots* can have small variations due the list of goals created by the generators with an additional source of randomness.

Tabla 7.10 shows the list of features sorted by type, as well as the position of the two feature filtering criteria. Both filters discarded features, 2 features in the case of GR or 23 features in the case of CFS. GR provides similar values (between $0.11 - 0.12$) in a group of 16 features, most of them from the SAS$^+$ formalism. The resulting rankings emphasize the importance of the fact balance, heuristic value and landmark graph

Table 7.8: List of useful features in *Barman* domain with LAMA-2011 planner, the ranking of the features, their gain ratio (GR) and the ranking provided by correlation based feature selection (CFS).

Type	Name	Ranking	GR	CFS
SAS – CG	Total edges	18	0.00	-
	Maximun output edge HV	17	0.00	-
	Maximum input weight HV	14	0.00	-
	Maximum output weight HV	13	0.00	-
SAS – DTG	Total edge	12	0.17	1
	Total weight	11	0.17	-
	Maximum input weight	15	0.00	-
	Maximum output weight	16	0.00	-
FB	Balance ratio	1	0.27	2
	Balance distorsion	6	0.26	3
Heuristic	Additive	5	0.26	4
	Causal Graph	4	0.26	-
	Context-enhanced Additive	7	0.26	-
	FF	9	0.26	5
	Landmark-cut	10	0.2	6
	Hred-black	3	0.26	-
Landmark	Number edges	8	0.26	-
	Maximum input edges	2	0.26	-

features over the rest. CFS only selects 12 features, selecting the SAS$^+$ formalism in the first case, fact balance, heuristic values and the last one from the landmark graph.

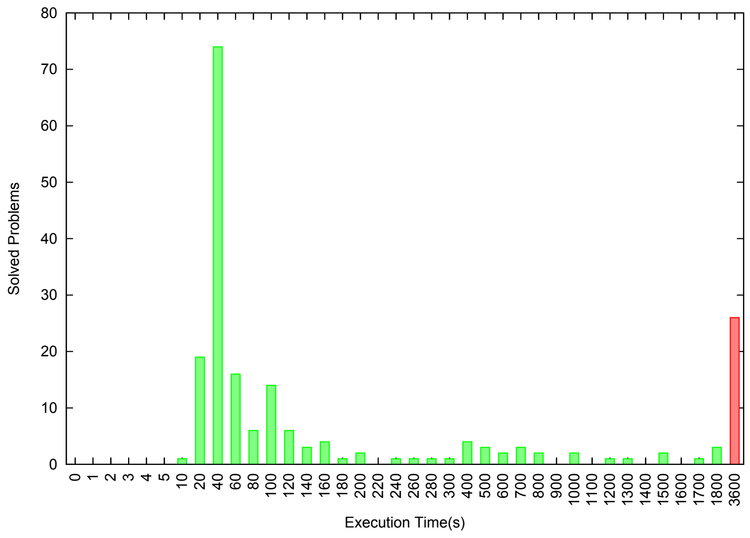
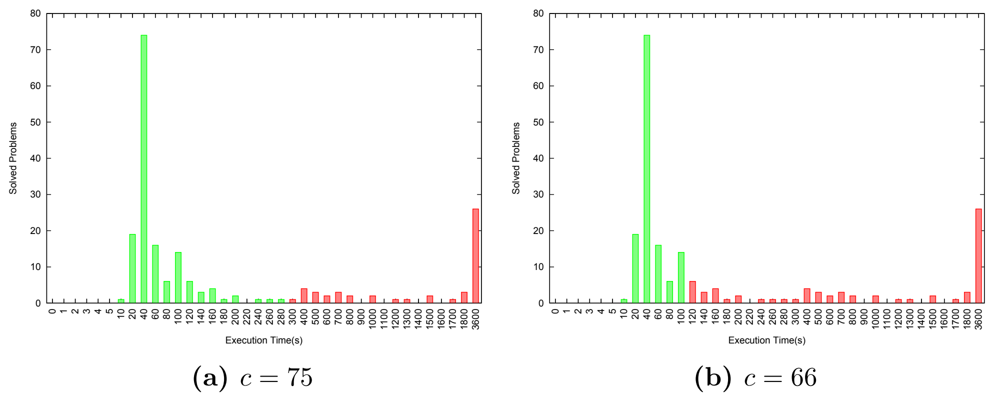


Figure 7.8: Execution time for the 200 problems of *Depots* domain with MERCURY planner.

Table 7.9: *Depots* Results in terms of accuracy (Acc) and area under the ROC curve (AUROC) with a original data and a cut-off of 75 and 66 with MERCURY planner.

	Original		75%		66%	
Algorithm	Acc	AUROC	Acc	AUROC	Acc	AUROC
ZeroR	87.0	0.45	75.0	0.50	66.0	0.48
J48	85.5	0.54	71.0	0.58	76.5	0.70
NaiveBayes	72.5	**0.84**	76.5	0.85	67.0	0.76
RandomForest	87.5	0.79	96.0	**0.98**	72.5	0.76
RotationForest	87.5	0.78	96.5	**0.98**	71.0	**0.78**



(a) $c = 75$ (b) $c = 66$

Figure 7.9: Execution time for the 200 problems of *depots* domain when 75% and 66% are label as solved problem (True) in green.

Table 7.10: List of useful features in *Depots* domain with Mercury planner, the ranking of the features, their gain ratio (GR) and the ranking provided by correlation based feature selection (CFS).

Type	Name	Ranking	GR	CFS
PDDL	Goal	17	0.12	-
FD	Translator task size	24	0.12	-
SAS - CG	High level variables	23	0.12	-
	Maximum output weight	27	0.11	1
	Average output edge HV	21	0.12	-
	Standard deviation output edge HV	22	0.12	-
	Average input weight HV	20	0.12	-
	Standard deviation input weight HV	26	0.12	-
	Average output weight HV	25	0.12	-
	Standard deviation output weight HV	19	0.12	-
SAS - DTG	Maximum input edge	15	0.12	-
	Maximum output edge	30	0.09	-
	Maximum input weight	31	0.09	-
	Average input weight	8	0.18	2
	Maximum output weight	16	0.12	-
	Average output weight	14	0.12	-
FB	FF ratio	28	0.10	-
	Balance ratio	17	0.12	3
	Balance distorsion	13	0.14	4
Heuristics	Additive	10	0.17	-
	Causal graph	1	0.34	5
	Context-enhanced additive	5	0.21	6
	FF	3	0.21	7
	Goal count	2	0.24	8
	Landmark count	9	0.18	9
	Landmark-cut	29	0.10	-
	Max	35	0.00	-
	Hred-black	6	0.20	-
Landmarks	Landmarks	12	0.14	-
	Number edges	7	0.18	10
	Father nodes	33	0.05	-
	Children nodes	4	0.21	11
	Nodes between	11	0.16	-
	Maximum input edges	34	0.00	-
	Maximum output edges	32	0.08	12

7.3.5 Depots Domain - Lama-2011 Planner

The next experiment is with the same domain, *Depots*, but changing the planner to LAMA-2011. The time allocated to solve the problems is 162.85 seconds on average. Figure 7.10 shows the progression of 200 problems in 1,800 seconds. The majority of the problems are solved within 160 seconds (139 problems). The problems start to be solved at 20 seconds from the execution and there are 21 problems with no solution (10.5%).

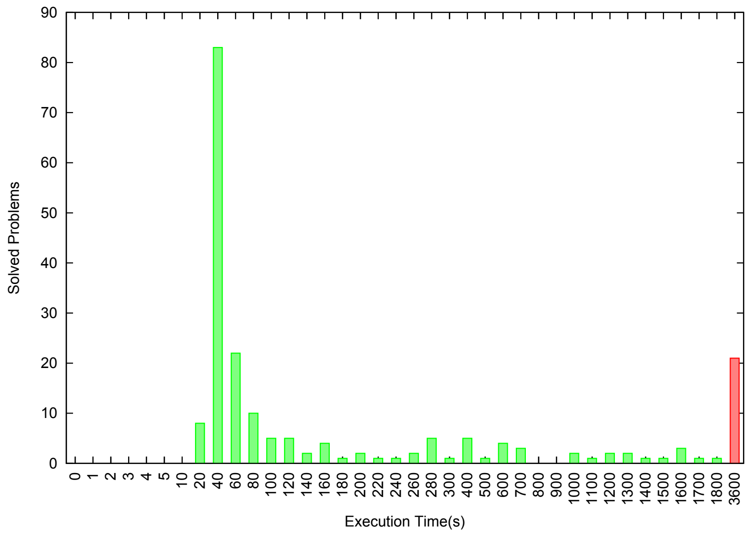
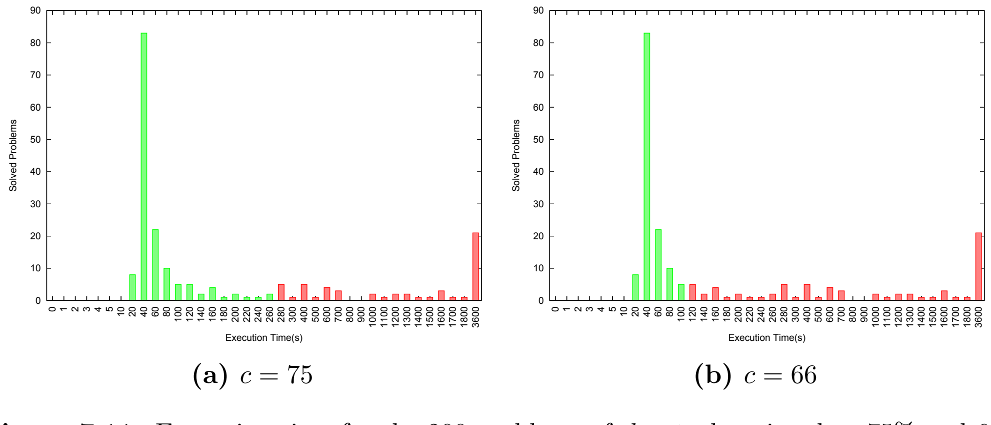


Figure 7.10: Execution time for the 200 problems of *Depots* domain with LAMA-2011 planner.

Figure 7.11 shows the datasets with different proportions of unsolved problems. Subfigure 7.11a labels them as unsolved after 280 seconds, and Subfigure 7.11b labels them as unsolved after 120 seconds.



(a) $c = 75$

(b) $c = 66$

Figure 7.11: Execution time for the 200 problems of *depots* domain when 75% and 66% are label as solved problem (True) in green.

Table 7.11 shows the results of accuracy and AUROC of the all datasets. Rotation Forest is the best algorithm in 2 of 3 cases, obtaining AUROC values of more than or equal to 0.75. The base algorithm obtains 0.45 in terms of AUROC, a value quite distant from the other algorithms.

Table 7.11: *Depots* Results in terms of accuracy (Acc) and area under the ROC curve (AUROC) with original data and a cutoff of 75 and 66 with LAMA-2011 planner.

	Original		75%		66%	
Algorithm	Acc	AUROC	Acc	AUROC	Acc	AUROC
ZeroR	89.5	0.45	75.0	0.50	66.0	0.48
J48	90.0	0.69	71.0	0.63	66.5	0.59
NaiveBayes	60.5	0.67	69.0	0.73	65.5	0.72
RandomForest	90.0	0.76	76.5	0.76	72.5	**0.75**
RotationForest	93.5	**0.80**	79.5	**0.78**	70.5	0.74

The next step is to discover the relevant features for the *depots* domain in the LAMA-2011 planner. The pre-selected features are the same as in the previous subsection. Table 7.12 shows the relevant features using the two criteria, GR and CFS. GR discarded 7 more than features, such as the maximum input and output edges of the landmark graph, h_{max} and h_{L-cut} heuristics. CFS only selects 5 features from the initial set, in contrast to the 12 selected in the MERCURY dataset.

The GR ranking values between MERCURY and LAMA-2011 are similar in relation to the type of feature, highlighting the features based on heuristic values and the landmark graph. In this domain, the CFS method always emphasizes *Balance Ratio* and the *Balance Distorsion* and discards features from PDDL and FD. These results are in accordance with the previous ones from the *Barman* domain, in which these features were discard in the first step.

Table 7.12: List of useful features in *Depots* domain with LAMA-2011 planner, the ranking of the features, their gain ratio (GR) and the ranking provided by correlation based feature selection (CFS).

Type	Name	Ranking	GR	CFS
PDDL	Goal	22	0.02	
FD	Translator task size	25	0.04	
SAS -CG	High level variables	26	0.03	
	Maximum output weight	28	0.04	
	Average output edge HV Avg	24	0.04	
	Standard deviation output edge HV	19	0.04	
	Average input Weight HV	23	0.04	
	Standard deviation input Weight HV	27	0.04	
	Average output weight HV	20	0.04	
	Standard deviation output weight HV	21	0.04	
SAS - DTG	Maximum input Edge	35	0.00	
	Maximum output edge	17	0.05	
	Maximum input Weight	30	0.00	
	Average input Weight	16	0.06	
	Maximum output weight	18	0.05	
	Average output weight	31	0.00	
FB	FF ratio	10	0.12	
	Balance Ratio	14	0.08	1
	Balance Distorsion	8	0.13	2
Heuristics	Additive	15	0.08	
	Causal graph	1	0.20	3
	Context-enhanced additive	4	0.15	4
	FF	5	0.14	
	Goal count	11	0.12	
	Landmark count	3	0.15	
	Landmark-cut	29	0.00	
	Max	32	0.00	
	Hred-black	6	0.13	
Landmarks	Landmarks	2	0.17	5
	Number Edges	7	0.13	
	Father Nodes	12	0.09	
	Children Nodes	13	0.09	
	Nodes between	9	0.13	
	Maximum input	34	0.00	
	Maximum output	33	0.00	

7.3.6 Floortile Domain - Probe Planner

The next experiment is in the *Floortile* domain using PROBE. This domain has the sixth highest value in terms of the coefficient of variation. Furthermore, this domain is quite difficult for this planner because it only solved 12 of the initial 30 problems.

This domain consists of a set of robots that use different colors to paint patterns on floor tiles. The robots can move around the floor in four directions (up, down, left and right). Robots paint with one color at a time, but can change their spray guns to any available color. However, robots can only paint the tile that is in front (up) and behind (down) them, and once a tile has been painted no robot can stand on it. An example of a floortile problem appears in figure 7.12.

For this set, robots need to paint a grid in black and white, where the color of the cell is always alternated. This particular configuration makes the domain hard because robots can only paint the tiles in front of them, since painting the tiles behind make the search to reach a dead-end in most cases. The problems have a 5x4 grid with two robots and two colors.

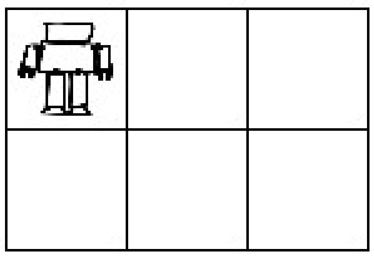


Figure 7.12: Example of Floortile domain in a grid of 3x2 with one robot.

Figure 7.13 details the problems solved in 1,800 seconds with PROBE. This planner only solved 56 of the 200 problems. Our procedure 7.2 does not generate a dataset of different proportions, and only the real proportion is considered as the percentage of unsolvability is 72%.

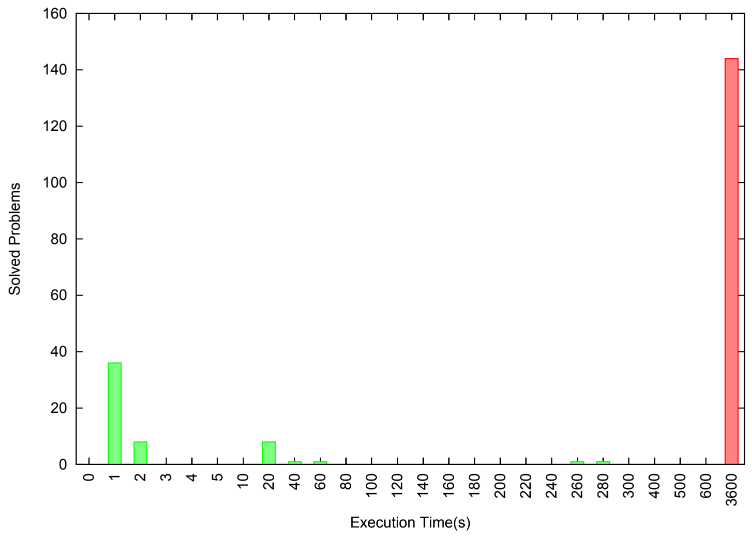


Figure 7.13: Execution time for the 200 problems of *Floortile* domain with PROBE planner.

Table 7.13 shows the results in terms of accuracy and AUROC. All of the algorithms are slightly better than the base algorithm ZeroR. However, these values are close to the performance of a random classifier.

Table 7.13: *Floortile* Results in terms of accuracy (Acc) and area under the ROC curve (AUROC) with original data with PROBE planner.

	Original	
Algorithm	Acc	AUROC
ZeroR	72.0	0.47
J48	72.5	**0.54**
NaiveBayes	63.0	0.50
RandomForest	60.5	0.51
RotationForest	70.5	0.52

As regards the relevant features, only 11 features have a standard deviation if more than one point. Table 7.14 shows that only the Balance ratio is relevant in the *floortile* domain. The GR method does not give any value to the rest of the features and as well like CFS, that also discarded these features.

Table 7.14: List of useful features in *Floortile* domain with PROBE planner, the ranking of the features, their gain ratio (GR) and the ranking provided by correlation based feature selection (CFS).

Type	Name	Ranking	GR	CFS
FB	Balance ratio	1	0.10	1
	Balance distorsion	3	0.00	-
Heuristics	Additive	2	0.00	-
	Causal graph	5	0.00	-
	Context-enhanced Additive	4	0.00	-
	FF	11	0.00	-
	Landmark-cut	10	0.00	-
	Hred-black	6	0.00	-
Landmarks	Landmarks	9	0.00	-
	Number edges	8	0.00	-
	Number father Nodes	7	0.00	-

7.4 General Feature Analysis

In this chapter, we propose to study the performance of the individual planners in homogeneous problem sets. We have analyzed the quality of the models and the relevant features in a combination of three domains and three planners. The datasets of interest were selected by considering the value of the coefficient of variation in the performance data evaluated.

The results have shown that the relevant features depend on the domain, which are not always are the same. Nevertheless, there is a sub-group of features that always appear in the three domains. This group consists of 9 features, 2 come from the Fact Balance (Balance Ratio and Balance distortion), 6 from the heuristic values (Additive, Causal Graph, Context-enhanced Additive, FF, Landmark-cut and Red-black) and 1 based on the landmark graph (Number of edges in the graph). What these groups of features have in common is that they are scattered among the problems, and consequently when they produce different performance data, the models are able to recognize these differences.

Figure 7.14 shows the importance of these features from between 0 to 1. These values are calculated using the values obtained from the GR between the maximum values in each dataset. This graph does not represent all of the significant features but all of them are common in the previous experiments.

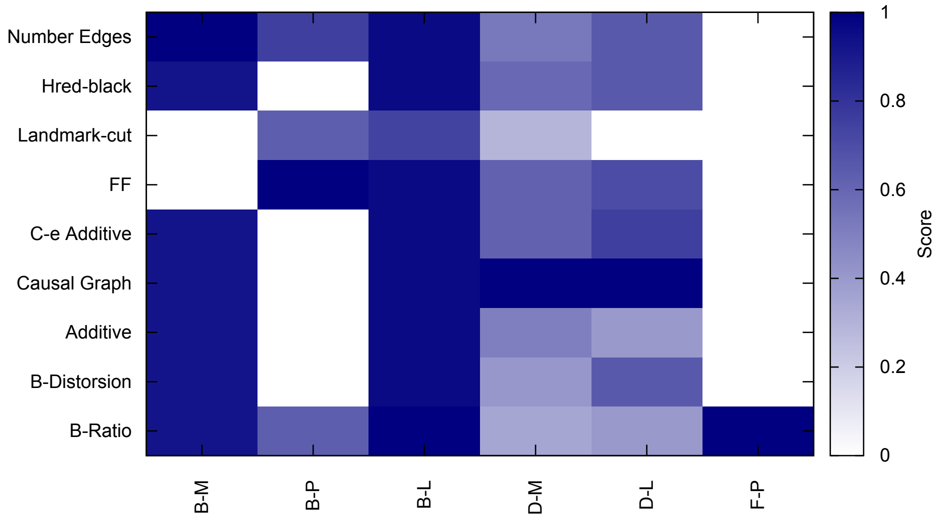


Figure 7.14: Selection features per the selected domains with the three planners. The x axis represents the different datasets and the y axis the list of features and the intensitivity of the color the relevance of the feature.

The x-axis represents the datasets, the first letter symbolises the domain B being the *barman* domain, D is *depots* and F is *floortile*. The second letter symbolises the planner, M being MERCURY, L is LAMA-2011 and P is PROBE. The most important feature is the Balance ratio ($B - Ratio$) because it has meaningful gain ratio in all datasets. The Causal Graph heuristic is also important because it has the best gain ratio when it appears as relevant for a dataset. Other important features are the Balance Distortion or Number in Edges in the landmark graph, since they give information in almost all cases.

In general, the features extracted from the relaxed plan give us about a greater illumination as to what really matters in the characterization of the problem. These features, in addition to the heuristic values of the initial state and the landmark graph could be crucial for knowing when a problem will be hard for a planner.

7.5 Summary

The majority of the state-of-the-art features produce the same or similar values in homogeneous problem sets. Therefore, they do not add discriminant information to recognize which tasks are potentially difficult for a given planner. In this chapter, we propose a new experimental procedure to evaluate a homogeneous problem set to confirm that our EPM gives a different representation per problem and that it improves the prediction capability. Several different ideas arise from this study: the relevance of each feature is not dominant across different domains and planners; however this set of new features give vital information to the EPM. The trained models in this case are far from perfect classifiers, however they demonstrated a better performance than random classifiers in almost all cases.

8

Temporal Approximation

State-of-the-art planning EPMs mainly focus on classical planning, like described in previous chapters in this dissertation. The features and learning processes followed are, therefore, adapted to that scope, and do not guarantee any prediction on the planner's performance in more expressive planning paradigms, like those paradigms in temporal planning, which are more expressive as regards the time constraints of the problems and domains, as it was described in the state in the art at Section 2.2.

The temporal problem incentive is to extract more representative features that represent temporal constraints, but there are no features in the current state of the art. Furthermore, some of the features previously used in classical planning might not be available in temporal planning. This type of planning usually requires the time constraints to be dealt with; actions have a duration and it might be necessary to run them concurrently to achieve some goals.

In this chapter, we propose a new set of features which are specific to problems dealing with durative actions and temporal constraints. We integrate them with existing classical planning features, and investigate the efficacy of EPMs in predicting the performance of temporal planners. In particular, we study the exploitability of features for building both classification and regression EPMs for a range of temporal planning engines, and we show the effectiveness of the models built for algorithm selection (Rice, 1976). Moreover, our analysis allows to gain some insights to be gained into the state of the art of temporal planning systems and we fill the gap between classical and temporal planning. This approximation differs slightly from the general formulation of the IBaCoP portfolios. The experimentation of this section is influenced by the limited research into the EPM of the planners in temporal planning. This is the first temporal portfolio approximation and temporal EPM could be interesting because these planners are good in different domains.

This chapter is organized as follows. First, we detail the planners and the benchmarks. Then, we explain all of the features used in this section (not all of them are the same as Chapter 4) and the time and process to extract them. After that, we present the results in terms of EPM, the results in a planning portfolio, and the planner selection. Finally, we present the conclusions of the temporal approximation.

8.1 Planner Selection

Planning systems able to deal with temporal problems are not as numerous as classical planning solvers. We initially collected the temporal planners which took part in the IPCs, that showed good coverage performance. We then ran them on our training instances, and removed those characterized by very poor performance (in terms of coverage). For this reason, we initially considered discarding three different planners (CPT4, LMTD and TLP-GP). Therefore, there are 8 planners considered and they are explained in Appendix A.3: LPG-TD,POPF2, YAHSP2, YAHSP2-MT, TEMPORAL FAST DOWNWARD, ITSAT, YAHSP3 and YAHSP3-MT.

8.2 Benchmarks

We consider temporal planning problems gathered from the temporal tracks of the last editions of the IPC, namely 2002, 2004, 2006, 2008, 2011 and 2014. Table 8.1 presents the training and test benchmarks used. The training set is IPCs 2008 and 2011: 25 domain models and 630 problems. It should be noted that where possible, we also considered different encodings of the same domain. Problems not solved by at least one planner were not included in the training set. The test set is IPCs 2002, 2004, 2006 and 2014. These IPCs are split into three parts: a set of new problems in known domains, and a set of new problems in unknown domains and the last IPC, which are detailed in Table 8.1.

Table 8.1: Domains included in the training and test phase categorized by origin. The origin of all problems and domains came from International Planning Competition, in the proper column the year of the competition appears.

Type		Origin	# Domains
Training phase		2008 – 2011	25
Test phase	Domains included at Training Set 2014	[2002 : 2006] 6	7
	Domains not included at Training set 2014	[2002 : 2006] 4	23

We considered different encodings of the same domain (for example, a domain could use the STRIPS representation , other numerical constraints, and other ADL features). Specifically, the test set that is included in training set only has three domains with different encodings (*Openstacks*, *Storage* and *Satellite*).

8.3 Features

We introduce 71 new features that are specific to temporal planning problems. This, together with several "classical" features provides us with 139 features in total. Overall, those features considered can be divided into six groups, described bellow.

Propositional PDDL

We consider 8 features that are extracted by considering both the domain and problem models that are specified in PDDL. They are a subset of the features proposed in previous works (Roberts et al., 2008), namely: the number of requirements in the domain, number of types, objects, predicates, facts in the initial state, number of (non-durative) actions and axioms. These features can be extracted from classical planning problems and thus are not temporal-specific. These features are explained in detail in Section 4.3.1.

Temporal PDDL

This set of features considers PDDL elements that appear in temporal models and numerical planning. For instance, we consider the number of assignments in the initial state, the presence of functions as the duration of the actions, the minimum, maximum, average and the standard deviation of the arity of the functions that are included in the actions, the number of preconditions and effects that should be fulfilled at the start, in the end or during actions execution (`at_start`, `at_end` and `over_all`). Intuitively, this set of features allows some insights to be gained into the constraints of the temporal part of the problem. For instance, it is possible to extrapolate whether some actions can be run in parallel (effects that are available early, preconditions that are required `at_end`), or actions are "locking" resources (preconditions that need to be fulfilled `over_all`). For these reasons, this set of features is deemed to be the most informative for EPMs aiming at predicting the performance of temporal planners. A total of 28 features are considered in this class and appear in Table 8.2.

Table 8.2: Temporal PDDL Features

Name	Description
assignment	Number of numeric assignments in the problem.
num durative actions	Number of durative action included in the domain definition.
type numeric duration	Number of durative actions have numeric duration.
type function duration	Number of durative actions have a function for the duration.
average numeric duration	Average, minimum and maximum of the value of the durative actions.
functions	Number of functions included in the domain definition
avg arity	Average, minimum and maximum of the arity of the functions included in the domain.
at start precondition	Average, minimum, maximum and standard deviation of `at_start` preconditions.
over all precondition	Average, minimum, maximum and standard deviation of `over_all` preconditions.
at start postcondition	Average, minimum, maximum and standard deviation of `at_start` postconditions.
at end postcondition	Average, minimum, maximum and standard deviation of `at_end` postconditions.
durative actions	Number of actions that included temporal restrictions.

General SAS$^+$

We consider features that can be extrapolated from the SAS$^+$ encoding (Bäckström and Nebel, 1995) which in contrast to the predicate-centric PDDL encoding is object-centric. For instance, the number of vertices and edges of the causal and domain transition graphs, which represent relationships between actions and objects, are considered. In total, 49 features belong to this class. It should be noted that not all of the features are temporal-specific, so they have also been explained in Section 4.3.3.

Problem Size

We have included several from SATzilla (Xu et al., 2008), however the ITSAT (Rankooh et al., 2012) planner does not provide a proper file to extract the features of the system. This process only extracts the first category of these features, called *"Problem Size Features"*. A total of 13 features are considered in this class and they are explained in Table 8.3.

Table 8.3: Problem Size Features. These features are from the 1-12 features SATzilla.

Name	Description
Ratio of Relevant Actions	Number of the final actions between the number of initial Actions.
NAction	Number of final actions
N Propositions	Number of all propositions usefull included
N Relevant Actions	Number of actions are included before simplification with SATElite.
N Relevant Propositions	Number of proposition that are included in the relevant actions (without instanciate).
Variables End	Created Variables at SATElite. formulation.
prop End	Number of proposition that are included in the instantiated actions (without instantiate).
action End	Instantiated actions a SATElite formulation post simplification.
Total Mutex Clauses	Number of mutex clauses in the variables.
ratio End	Ratio of variables to clauses
TEvent Clauses	Number of clauses in original formula.
TClauses	Number of simplification clauses.
Number Files	Number of the temporary files created.

Temporal Fast Downward

The pre-processing of Temporal Fast Downward (Eyerich et al., 2012) uses multi-valued state variables and the handling of logical dependencies and arithmetic subterms via axioms. The values of these numerical variables are set directly by the actions or, in the case of compound expressions, determined by newly introduced numerical axioms. Comparisons between the numerical expressions are translated to logical variables whose values are determined using comparison axioms in the internal process of the planner. This is a hard process, in which the system counts the number of axioms. Furthermore, other considerations are included in this process to compile the temporal restrictions with auxiliary variables. For example, the final state variables in the SAS$^+$ formulation are divided into those affected by temporal operators and derived variables (these variables are computed using axioms). Several of these features are previously included in the general process in FD, however it is not possible to calculate others in TFD. A total of 11 features are considered in this class, and they are explained in Table 8.4.

Name	Description
init	Number of predicates that appear in the initial state
goals	Number of predicates that appear in the goal state.
function administrator	Auxiliary number of TFD
final queue length	Size of the queue in the translation process.
translator operator	Number of operators that appear in the translation process.
necessary operators	Number of operators at preprocess phase.
uncovered facts	Number of facts included in the preprocess phase.
necessary variables	Number of variables that appear in the translation process.
relation axiom	Number of axioms that are relational in TFD.
functional axiom	Number of axioms that are functional in TFD.
true axioms	Number of axioms that are true in the translation process.

Table 8.4: TFD Features

Temporal SAS$^+$

This group of features is a extension of the Temporal Fast Downward features. In a sense, these SAS$^+$ related features are created based on the internal structure of the planner to create the temporal planning task. Several are specific to the aspects of the planners to treat fluents and handle numerical features, because these characteristics are not supported by FD, and this system is built on top of it. Consequently, several of them have no semantic meaning, only for the translation process from temporal PDDL to SAS$^+$. There are 30 features which are listed in Table 8.5.

8.4 Extracting Features

We observed that Propositional PDDL features require negligible time to be extracted, while temporal PDDL feature extraction requires at most 10 CPU time seconds. Furthermore, extracting SAS$^+$ features is usually more expensive; on average, close to 29 seconds are required. It is something we also noticed in a few problems, we considered

Table 8.5: Temporal SAS+ Features

Name	Description
durative actions	Number of durative actions identified by TFD.
action counter	Number of different actions when the temporal task are translated a general SAS$^+$
function symbols	Number of symbols of the TFD.
generated rules	Number of rules generated in TFD in the translation process.
final queue	Number of the elements that appear in the planning queue.
translator variables	Number of temporal variables in TFD.
translator derived variables	Number of temporal derived variables in TFD.
translator facts	Number of temporal facts in TFD.
mutex key	Number of mutexes at the initial state
strips to sas	Number of auxiliary variables used in a temporal SAS$^+$
ranges	Number of different variables that are numeric and have different intervals.
goal list	Number of elements in the final state in the temporal task.
task init	Number of elements in the initial state in the temporal task.
translator durative act	Number of actions in a preprocess phase.
translator axiom	Number of axioms in a translation phase.
translator num axioms	Simplified axioms in a translation phase.
translator num axioms by layer	Number of actions per level
translator max num layer	Maximum number of layers
translator num axiom map	Number of total axioms that appear in all process
translator const num axioms	Minimum number of necessary axioms
translator reachable	Number of variables that are reachable in the initial state.
translator mutex group	Number of groups of variable/value pairs of which no two can be simultaneously true.
translator translation key	Auxiliary value of TFD
avg level	Average of the number of levels.
str level	Standard deviation of the number of levels.
global num type start	Number of transitions that are label at at_start.
global num type end	Number of transitions that are label at at_end.
global min level	Minimum number of levels in the graphs.
global max level	Maximum number of levels in the graphs.
global total level	Total number of levels in all graphs.

in the experimental analysis (*crewplannning*, *matchCellar* and *transport*). The extraction of the SAS$^+$ features needs a significantly longer CPU time. Only in a few domains the CPU time required was significantly higher. However, only in a few domains considered in our experimental analysis, the time required to extract these features was longer than 13 seconds.

Problem size features are cheaper than SAS$^+$ features, and usually require around **1** CPU-time second to be computed. It is worth noting that he SAS$^+$ features have not been computed, due to timeout or the memory running out, in approximately 15% of the problems considered in our experimental analysis. Table 8.6 shows the average and maximum time required to extract the different sets of features, as well as the percentage of problems in which the extraction was completed on time. A timeout of 100 seconds for the overall extraction process has been used. Using an excessively large amount of CPU time for extracting features reduces their usefulness, in the light of the fact that planners also tend to solve problems quickly, or not at all (Howe and Dahlman, 2002).

Propositional and temporal PDDL features are extracted using different systems, and therefore they are independent. Both general, temporal SAS$^+$ and TFD features are extracted using TFD, thus their extraction time is the same.

Table 8.6: Average and Maximum CPU time needed to extract features, the number of features per group (#) and the percentage of successful features extraction (succ.).

		Average	Maximum	#	Succ.
PDDL	Prop	0.01	0.15	8	100%
	Temp	5.06	10.00	28	100%
P. size		0.89	2.00	13	80%
SAS$^+$		28.89	50.00	90	80%
Total		33.96	60.15	139	-

Finally, many useful methods exploited for computing classical planning features are not able to handle temporal planning problems, since they rely on the modules of classical solvers.

For similar reasons, we could not extract any useful SAT-related feature, although there is a planner – ITSAT (Rankooh et al., 2012) – that exploits SAT for solving temporal planning problems, however, the CNFs generated only represent the variables, not the problem.

They can be naively solved by unit propagation, and thus their ability to provide information is extremely limited. For this research, we considered the well known features for Torchlight (Hoffmann, 2011). However, it does not support temporal constrains and these features are computationally expensive. For these reason, we discarded it.

The translation process from PDDL to SAS$^+$ is in the TFD system. It is quite similar to the general process of IBaCoP planners (changing FD to TFD). This algorithm is included when the features are extracted according to the information required. PDDL features (temporal and prepositional) and the problem size features only require PDDL files. SAS$^+$ (temporal or general) and Temporal Fast Downward features require the problem in a finite-domain representation. The translation process is performed over several stages.

8.5 Experimental Settings

All planners and feature extractors were run on a cluster with Intel XEON 2.93 Ghz nodes with 8 GB of RAM each, using Linux Ubuntu 12.04 LTS. The planners had a cut-off of $1,800$ CPU-time seconds, and a maximum of 4 GB of RAM, like in the temporal track of IPC-2014. The features extraction cut-off time was set to 100 seconds. A maximum of 4 GB of RAM for feature extraction was also used.

We considered EPMs for both *classification* and *regression* approaches. Given a planner, classification approaches classify planning problems into a single category, corresponding to the fact that the planner will either solve the problem or not. Regression techniques model the behavior of each planning engine in order to predict its runtime on a given problem.

Firstly, we assessed the performance of various classification and regression models (45 different algorithms in total), using the WEKA tool (Hall et al., 2009). We considered linear regression, neural networks, Gaussian processes, decision trees, regression methods, support vector machine and rule-based techniques. The EPMs generated were evaluated using a 10-fold cross-validation approach on a uniform random permutation of the training instances. It is a standard method in which nine slices are used for generating the model, and the tenth for assessing the performance of the model as explained in Section 4.1. Furthermore, we carried out other interesting tests divided into three groups to understand how well these models are generalized as detailed in Table 8.1.

8.6 Experimental Results

In order to evaluate how different features affect the ability to predict the planners' performance, we consider 6 distinct groups of features. The features have been grouped according either to the encoding they refer to, or their temporal-specificity, as follows. We also considered a small set of automatically selected features.

All indicates the whole set of computed features (139). **PDDL** refers to the 49 Features including Propositional and Temporal PDDL. **SAS$^+$** considers the 90 features that are extracted by considering the SAS$^+$ encoding. 68 **nT** (Non-Temporal) features are typical of classical planning. The features are gathered from Propositional PDDL and General SAS$^+$ sets. **T** (Temporal) this set considers the 71 features that are extracted by considering Temporal PDDL and Temporal SAS$^+$ encodings. Finally, the **Sel** set includes 11 features, gathered from all of the sets considered, that have been identified through a feature selection process.

Feature selection is made by looking at a J48 decision tree (Quinlan, 1993), which is built for predicting the solvability of the training instances, by considering planners as input information. Given the model, we select the features used in nodes placed in the top fifth level of the decision tree. They are believed to be important since, according to the J48 algorithm, they provide the best information gain (Quinlan, 1993). This can be seen as a supervised method of feature selection. Moreover, considering the top nodes allows potential overfitting to be avoided; it can arise in the lower-level leaves of the tree that are used for classifying a very few instances. The accuracy of the EPM generated by the J48 algorithm is good, with almost 91% of accuracy. Therefore, we believe information extracted from this model is relevant.

According to the process described, there are 11 features selected, distributed as follows in Table 8.7.

Table 8.7: Feature selection at Temporal Domains

Type	Name
Propositional PDDL	predicates
Temporal PDDL	type function duration
	avg arity
	min at start precondition
	max over all precondition
	min at end postcondition
	max at end postcondition
Problem Size	ratioEnd
General SAS$^+$	max outputEdgeCG
	max inputEdgeDTG
Temporal SAS$^+$	translator durative act

Temporal features tend to appear earlier in the J48 decision tree, thus they are deemed to be more informative. On the other hand, this distribution of selected features across the SAS$^+$ and PDDL sets, requires that both the extraction processes are executed, and successful.

8.6.1 Classification

In classification, a different predictive model is built per planner, in order to predict whether the planner will find a solution to a given problem or not. Rotation Forest (Rodriguez et al., 2006) performed the best from among the classification approaches considered in the training instances, and is exploited hereinafter.

Table 8.8 shows the results in terms of per-solver accuracy. The performance in training instances is very good, regardless of the set of features considered. Usually, any set of features allows around 90% of accuracy to be achieved. This is probably due to the fact that each class has a number of informative features, and that some domains have a large number of problems. It should also be noted that the two classes considered (solved, unsolved) are balanced between all the planners; the maximum difference is $40 - 60\%$. In order to achieve this class balance, we assessed the initial distribution between classes and, in imbalanced cases, randomly over-sampled the minority class. This approach is common practice in machine learning (He and Garcia, 2009).

The results at the top of the Table 8.8 clearly indicate this in training instances, EPMs are able to identify relevant features, and combine them for predicting the solvability of problems. The results at the bottom of the table are the test from IPC-2014. The performance on testing instances provides a number of interesting insights: (i) the PDDL set usually leads to the best prediction results. Even when it is not the best set, it is very close to the best one; (ii) using either a temporal or non-temporal set of features allows similar prediction results to be achieved; (iii) counter-intuitively, using all the features together does not guarantee the best performance; (iv) TFD and YAHSP3 behaviors are hard to predict in testing instances, and (v) the set of selected features usually allows EPMs to achieve good prediction results, particularly considering that only 11 features are considered for a domain-independent prediction. We observed that TFD and YAHSP2/3 show a very different behavior on training and testing problems, possibly because of the new domains and/or significantly larger instances used in the testing set. TFD translates the PDDL planning problem into SAS$^+$, and then solves the SAS$^+$ problem; the translation phase can be slow and, sometimes, requires a huge amount of memory. On large instances, such as those used in the IPC-14, it happens that the translation step fails due to the lack of memory available; this is clearly hard to predict for an EPM that has been trained on smaller instances, where this issue did not usually arise. Moreover, both planners have some issues in dealing with problems that need concurrency to be solved. Another reason for this behavior is that the planner is able to find plans requiring some limited from concurrency point of view. Furthermore, it is necessary to consider that this planner translates the task and this process fails. In fact, on the benchmarks of IPC 2014, TFD is not able to solve problems from 5 domains, while YAHSP3 is not able to solve from 3 domains. In the results from the last IPC, TFD was unable to solve any problems in 5 different domains.

Finally, with regard to the fact that the set considering all the features is not always the best option, we believe this is mainly because of introduced "noise". Our hypothesis is supported by the results achieved using the 11 features selected: they represent a (hopefully) noise-free set of features, and their exploitation allows results close to those achieved using the **All** set to be obtained. It should also be noted that the sets considered have some overlap, and this partially explains why in some cases they show a similar performance.

Table 8.9 shows the results from other domains. The first part of the table details the known domains. The performance results of the planners are a lot worse than the previous test instances (IPC-2014). One of the possible reasons for the results is that the encoding of the known domains are not supported by the new planners, and the accuracy of the performance modelling decreases because the "input features" are similar, but the result are totally different. From these results, it could be highlighted that the YASHP2 and YASHP3 versions provide similar results in all cases. The TFD planner obtains similar results in previous test set, around 70% of accuracy taking into account that the planner has a worse performance in the training phase.

The only planner that achieves better results is ITSAT, which increases the performance in known domains, achieving 100% with the PDDL domain definition. In contrast, LPG-TD decreases the performance in terms of predictability, but the results of the planner are better than the training instances. This planner in the training phase

Table 8.8: Accuracy (higher is better) of classification EPMs predicting whether a planner will solve a problem or not. Results are shown for each considered set of features, on both training (upper table) and testing (lower table) instances. Performance on training are assessed through cross-validation. Bold indicates the best results, also considering hidden decimals.

Training Instances						
Planner	All	PDDL	SAS$^+$	nT	T	Sel
LPG-TD	92.6	88.5	88.6	**92.7**	91.9	88.4
POPF2	88.6	87.2	84.9	**88.7**	88.2	87.7
YAHSP2	89.6	91.0	89.1	87.9	89.9	**91.4**
YAHSP2-MT	**95.5**	91.9	89.3	93.9	95.3	89.8
ITSAT	**94.1**	88.2	88.4	93.6	94.1	89.1
TFD	94.1	87.5	84.9	93.5	**94.2**	88.8
YAHSP3	91.0	90.8	89.0	89.7	91.2	**93.1**
YAHSP3-MT	**93.9**	93.4	90.7	92.2	93.8	90.7

IPC-2014 Testing Instances						
Planner	All	PDDL	SAS$^+$	nT	T	Sel
LPG-TD	76.5	**81.5**	73.0	75.0	74.5	76.0
POPF2	**87.0**	77.5	83.5	86.5	80.5	68.5
YAHSP2	74.5	**76.0**	67.5	57.0	59.5	56.5
YAHSP2-MT	63.5	**80.5**	65.0	72.5	57.0	68.0
ITSAT	**89.0**	88.5	73.0	84.5	88.5	74.5
TFD	67.0	67.0	69.5	**71.0**	67.0	67.0
YAHSP3	60.0	**74.0**	61.5	59.0	57.0	56.0
YAHSP3-MT	75.0	**82.0**	73.0	65.5	57.0	78.0

has a solve/unsolved ratio of 329/301, almost perfectly balanced, but the test phase achieves 143/53 ratio in known problems and 359/375 in unknown domains, a relevant unbalanced group. In any case the percentage of solved/unsolved problems are totally different of the training phase.

In order to investigate how the importance of features varies between training and testing problems, we applied our selection process in EPMs built by considering the testing instances only. Like to the selection process made on the training problems, 11 features are selected.

One of them is exactly the same: the minimum number of effects that become true when the action finishes (`at_end`). Moreover, six features selected according to testing instances are strongly related to those extracted in training problems, as they consider similar aspects of the problem, but from slightly different perspective: the maximum arity of temporal functions (PDDL), minimum number of `at_start` preconditions (PDDL), minimum duration of an action (PDDL), standard deviation of incoming edges of the domain transition graph (SAS$^+$), number of variables (SAS$^+$), and number of relevant actions (SAS$^+$). Finally, the remaining features are completely different from those included for EPMs built considering the training instances. This is the case for: the number of PDDL requirements (PDDL), number of facts in the initial

Table 8.9: Accuracy of classification EPMs to predict whether a planner will solve a problem or not. Results are shown for each considered set of features as the same as previous table. Results of EPMs on different sets of features, on known domains (upper Table) and unknown domains (lower Table). Bold indicates the best results.

Know Domains	All	PDDL	SAS$^+$	nT	T	Sel
LPG-TD	42.5	**81.2**	33.3	31.2	76.3	53.8
POPF2	65.6	72.0	67.20	45.7	**77.4**	51.6
YAHSP2	43.6	75.8	74.19	76.9	**78.0**	78.0
YAHSP2-MT	**80.1**	57.5	76.9	76.3	79.0	79.0
ITSAT	97.3	**100**	76.3	86.0	98.4	92.5
TFD	42.5	39.3	71.5	**75.3**	44.6	41.4
YAHSP3	71.5	**77.4**	65.1	74.7	57.5	77.4
YAHSP3-MT	53.2	78.5	76.9	75.8	**78.5**	78.5

Unknown Domains	All	PDDL	SAS$^+$	nT	T	Sel
LPG-TD	62.6	48.9	**64.4**	55.4	55.7	56.8
POPF2	59.8	57.0	67.3	**77.1**	42.1	57.1
YAHSP2	48.9	80.3	**84.7**	84.5	73.2	75.7
YAHSP2-MT	75.0	71.4	79.4	**82.2**	74.7	72.0
ITSAT	92.4	86.2	90.0	75.0	89.3	**92.4**
TFD	57.6	53.7	**70.8**	68.0	40.8	30.6
YAHSP3	62.2	73.7	**78.2**	71.9	78.2	55.7
YAHSP3-MT	**86.4**	65.0	78.3	72.7	73.6	57.1

state (PDDL), ratio between the weight and the edges in the causal graph (SAS$^+$), and the ratio between edges and variables of domain transition graph (SAS$^+$).

Overall, also considering the results achieved by EPMs exploiting the **Sel** set of features, this analysis confirms their ability to inform. It also indicates that the technique we designed for selecting informative features is reasonably accurate, and selects features that allow EPMs to exploit them to generalise different benchmarks.

8.6.2 Regression

Regression EPMs predict the runtime a planner will need to solve a given problem. The runtimes of our planners vary from between 0.1 to 1,800 CPU seconds, for solved instances. For unsolved problems, we considered 2000 CPU time seconds. This was also done to allow for the regression EPMs to predict unsolved instances (when the predicted runtime is between 1,800 and 2,000 seconds) without adding too large a bias towards unsolved benchmarks. It should be noted that, otherwise, it would be impossible to distinguish between problems solved in approximately 1,800 seconds and unsolved runs. Given the large variations in CPU times, our regression models predict the log-runtime rather than absolute time to use them as a *Logarithm time* strategy (see Section 5.2). Experimentally, we observed that the Decision Tables algorithm (Kohavi, 1995) allows, –on average–, the most accurate predictive models to be generated.

Table 8.10 shows the results, in terms of the Root Mean Square Error (RMSE), of the best regression models with a 10-fold cross validation on a uniform random permutation of the 630 training instances, and on the IPC 2014 testing instances. Firstly, we noticed that runtime of the prediction algorithms is challenging, according to the RMSE values. On the other hand, RMSE is sensitive to the occasional large error (e.g., predicting an unsolvable instance that can be solved quickly), thus actual predictions can be better, on average. Futhermore, training instances are usually representative of testing ones, since they allow regression EPMs to achieve similar results to the IPC 2014 benchmarks.

As regards the different classes of features, using the **Sel** set often allows the best regression EPMs to be built. This set is noisy free and very informative–, while the **SAS$^+$** set does not provide informative features for predicting runtimes. Remarkably, we also observed that the features from the temporal set are very informative, and allow a prediction performance to be achieved which is usually very close to the best.

Interestingly, we noticed that the regression approach has a similar RMSE performance to the TFD planner in training and testing instances. This was not the case for the classification model. On the other hand, we observed that YAHSP-BASED systems show a very different behavior in training and testing performance than in the classification case. In particular, the behavior of MT versions is the most challenging to predict. This has a limited impact on the ability of the planner to solve instances, but makes the actual runtime harder to predict.

Table 8.10: Root mean squared error (lower is better) of regression EPMs built by using Decision Tables. A different model is built for each planner. Results of EPMs on different sets of features, on training (upper Table) and testing (lower Table) instances are shown. Bold indicates the best performance.

Training Instances						
Planner	All	PDDL	SAS$^+$	nT	T	Sel
LPG-TD	1.49	1.57	1.84	1.54	**1.48**	1.49
POPF2	2.12	2.27	2.53	2.23	2.11	**2.05**
YAHSP2	1.76	1.45	2.07	1.86	1.65	**1.27**
YAHSP2-MT	1.41	1.45	2.25	1.84	1.33	**1.30**
ITSAT	1.45	1.58	1.68	1.41	1.42	**1.38**
TFD	2.18	2.32	2.56	2.19	2.16	**2.02**
YAHSP3	1.61	1.60	2.04	1.81	1.43	**1.41**
YAHSP3-MT	1.42	1.28	1.29	1.55	1.21	**1.17**
IPC-2014 Testing Instances						
Planner	All	PDDL	SAS$^+$	nT	T	Sel
LPG-TD	3.29	3.56	2.61	2.60	3.44	**2.20**
POPF2	2.49	2.43	2.22	2.84	**2.48**	2.76
YAHSP2	2.76	2.55	3.22	2.76	**2.37**	3.63
YAHSP2-MT	**2.83**	3.05	3.36	3.08	2.89	2.86
ITSAT	2.06	2.28	2.54	2.42	2.36	**1.87**
TFD	2.51	2.73	2.87	2.80	2.83	**2.19**
YAHSP3	2.60	3.33	3.23	2.85	2.79	**2.20**
YAHSP3-MT	2.99	2.85	3.12	3.27	2.65	**2.64**

Table 8.11 shows the results in the same terms as previous table, but changing the test sets. The first part of the table corresponds to the known domains and the last part to the unknown domains. These results are totally comparable with the previous ones, in which case the worse predictability is LPG-TD. The classification results show a high number of solved problems and as a consequence, the predictive time changes in the same way. An overdone example of that, is a ZeroR model: a predictive time based on the average. If 51% of the data is unsolved, the predicted solution is $ln(2,000)$. For this reason, the predictive models tend to return low times (average time in the training phase) as the predictive model learned in the training phase however the real time and error is high.

Table 8.11: Root mean squared error of regression EPMs built by using Decision Tables. A different model is built for each planner. Results of EPMs on different sets of features, on Known domains (upper Table) and unknown domains (lower Table) .

Know Domains						
	All	PDDL	SAS	nT	T	Sel
LPG-TD	3.02	**3.02**	3.54	3.53	**3.02**	**3.02**
POPF2	2.86	2.46	2.53	2.67	**2.43**	2.46
YAHSP2	1.73	**1.57**	2.03	2.06	**1.57**	1.62
YAHSP2-MT	3.18	3.16	2.12	2.13	3.16	**1.54**
ITSAT	1.61	1.61	**0.87**	0.99	2.15	2.29
TFD	3.47	3.47	2.98	**2.94**	3.45	3.09
YAHSP3	1.46	**1.51**	2.12	2.12	1.47	1.57
YAHSP3-MT	2.42	1.99	1.95	1.97	1.68	**1.49**
Unknown Domains						
	All	PDDL	SAS	nT	T	Sel
LPG-TD	2.86	2.86	**2.13**	2.86	2.86	2.31
POPF2	2.26	2.35	2.15	**2.06**	2.35	2.36
YAHSP2	2.18	**2.15**	2.39	2.37	**2.15**	**2.15**
YAHSP2-MT	2.80	2.78	2.40	2.35	2.78	**2.02**
ITSAT	2.70	2.70	2.45	2.56	2.84	**2.85**
TFD	3.86	3.86	2.81	2.78	3.86	**2.80**
YAHSP3	2.15	2.15	2.46	2.44	**2.12**	2.32
YAHSP3-MT	2.13	2.03	2.16	2.20	2.02	**1.88**

8.6.3 Exploiting EPMs for Algorithm Selection

After evaluating the prediction performance of classification and regression EPMs, we compared them to understand which is the most promising approach to be used for on-line algorithm selection. In particular, we tested them to select the most promising planner to exploit a given (and previously unseen) testing instance. In our work, EPMs are built with the aim of maximising coverage, i.e., the number of problems solved. Algorithm selection made by exploiting the classification EPMs considers only planners which are predicted to solve the given problem. Among them, the planner with the best EPM accuracy (in training instances) is selected. Instead, regression EPMs predict

Table 8.12: Performance, in terms of coverage and overall IPC score, of regression and classification EPMs exploited for algorithm selection, the best basic solver according to coverage (Best-C), the best basic solver according to IPC score (S-Best), the virtual best solver (VBS), and a static portfolio including 4 planners (B4P). The rows in grey indicate the domains that are not included in the training set. Bold indicates the best performance.

	Classification				Regression				Best		VBS	B4P
Domain	All	Sel	nT	T	All	Sel	nT	T	C	S		
TMS	18	18	16	18	18	18	18	18	0	0	18	0
Turn&Open	12	12	14	15	17	17	17	17	0	0	17	15
Storage	17	17	17	17	17	17	17	17	17	9	17	17
Driverlog	7	2	6	0	13	0	13	13	13	9	13	12
Floortile	20	20	20	20	20	20	20	20	20	8	20	20
MatchCellar	19	20	20	20	20	20	20	20	0	0	20	20
MapAnalyser	10	14	9	10	7	7	7	7	7	20	20	20
RTAM	0	6	0	3	20	20	20	20	20	20	20	20
Satellite	12	3	6	2	20	20	20	20	20	20	20	20
Parking	14	20	20	20	20	20	20	20	20	20	20	20
Coverage	129	132	128	125	**172**	159	**172**	**172**	117	106	185	164
IPC-Score	91.8	102.4	95.1	105.8	**129.3**	126.6	**129.3**	**129.3**	62.1	86.2	185	72.5

the runtime of each planner; that predicted to be the fastest is selected. Table 8.12 shows the results, in terms of number of problems solved and the IPC speed score for classification and regression EPMs using different sets of features. In this experimental analysis, the IPC score as defined in the Agile track of IPC 2014 is used. For a planner $\mathcal{C}$ and a problem p, $Score(\mathcal{C}, p)$ is 0 if p is unsolved, and $1/(1+\log_{10}(T_p(\mathcal{C})/T^*_p))$, where $T_p(\mathcal{C})$ is the CPU-time needed by planner $\mathcal{C}$ to solve problem p and T^*_p is the CPU-time needed by the best considered planner. The IPC score for a set of problems is given by the sum of the scores achieved in each problem considered.

For the sake of comparison, Table 8.12 includes the performance of the virtual best solver (VBS) which represents an Oracle that always selects the best possible planner for solving the specific problem, the best two basic solvers according to (C)overage (LPG-TD) and IPC (S)core (YAHSP2), and a static portfolio (B4P). The portfolio includes the best four planners, according to coverage performance on testing instances: LPG-TD, YASHP2, YASHP3 and TFD. The Solvers are ordered according to their coverage (descending order) and each planner runs for 1/4 of the cutoff time (i.e., 450 seconds per planner). Considering these additional systems –VBS, B4P and the best basic solvers–, allows to get a better and more complete understanding of the performance of the algorithm selection through EPMs. Figure 8.1 shows the number of problems solved as, regards the CPU time, of the solvers considered.

Classification and regression EPMs allow better coverage results to be achieved than the best basic solver (+11% and +32.5%, respectively). Moreover, performance achieved using regression EPMs is very close to the Oracle performance, and better than using the static portfolio. It is useful to remember that B4P has been configured by considering the performance of the planners in testing instances, while both regression and classification EPMs have been trained on a different set of instances. From this

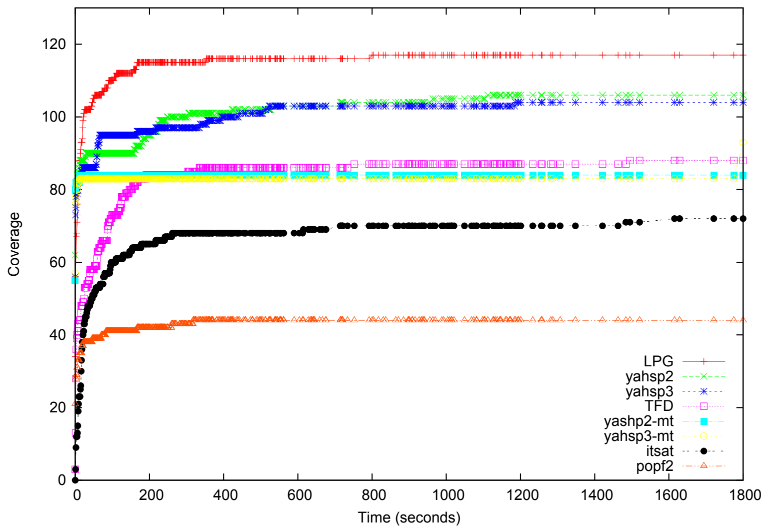


Figure 8.1: Performance, in terms of coverage over time, of the considered solvers on the benchmarks of the IPC 14 temporal track.

perspective, the proposed EPMs demonstrate their ability to generalize, since they provide useful predictions for make the algorithm selection in unseen instances. Although, we observed that regression EPMs are always better than classification models, both in terms of coverage and IPC score, this is probably due to the fact that classification EPMs do not estimate the performance difference between solvers, and a misclassified problem can easily go from success to failure. Regression techniques model the behavior of algorithms, which are ordered accordingly. In this way, a mistakenly selected planner usually leads to a longer execution runtime, however the selection of solvers with extremely poor performance is avoided.

As regards the sets of features considered, we noticed a very different behavior in classification and regression EPMS. Classification achieves the best coverage performance when using the selected set of 11 features; the IPC score on that set is close to the best, which is obtained using Temporal features. On the other hand, the **Sel** set is not the most informative for algorithm selection through regression; using the whole set of features –or even the set including only temporal / non-temporal features– allows a better performance to be achieved.

Table 8.13 shows the results, in terms of number of problems solved and the IPC speed score for classification and regressions EPMs using different sets of features. Furthermore, we included the result of the best planner component in terms of score and coverage, which is LPG-TD. This planner achieves better results than our EPM in terms of problems solved, but the scores in the regression strategies are always higher. The regression models are working better than classifications ones. These models not only try to predict whether a planner solves the planning task,but also try to predict which planner solves the problem soonest. This table has the results of the unknown domains taking into account that the unsolved domains are not discarded. Other domains are only close to being unsolved and several planners only solve a few of them, for example *UMLS_FLUENTS*, *UMLS_FLAWS*. In this case, all approximations that always select LPG-TD planner are worse in terms of coverage, however LPG-TD is not the best planner in all cases in the training phase. A selection base of the training phases always select the TFD planner, and this planner solves 230 problems. Six of our strategies get more than this value. These results entrench the idea of a portfolio of planners, with different characteristics to try to deal with unexpected situations. In addition, only *TANKAGE, Notankage, airport-s, pipesworld-MT, depots-simple-T, rover-Simple-T, driverlog-simple-T* and *zenotravel-simple-T* are easy to predict. In contrast, the remaining domains are very difficult in term of predictability, in which cases, only one planner is the correct answer. The real importance of these results is the correct predictability of every planner. If you have an unknown situation, deciding on a single planner without information could be a bad decision. In these cases, the EMPs are the best options, and our approximations guarantee the fastest planner for carrying out the temporal task.

Table 8.13: Performance, in terms of coverage and overall IPC score, of regression and classification EPMs exploited for algorithm selection at known domains.

	Classification				Regression				C & S	VBS	B4P
Domain	All	Sel	NT	T	All	Sel	NT	T			
TANKAGE	24	0	28	22	29	29	22	29	28	30	30
UMLS_FLUENTS	16	16	1	0	43	43	43	43	50	50	50
UMLS_FLAW	0	0	0	3	12	12	12	12	50	50	50
NOTANKAGE	17	0	24	27	23	23	23	23	30	31	31
Airport-S	32	39	39	30	36	36	36	36	34	46	44
Pispesworld-MT	28	21	25	22	29	29	28	29	26	30	30
Airport-adl	0	0	0	0	0	0	0	0	0	0	0
Rovers-MT	16	8	16	14	15	15	15	15	16	19	19
Trucks-adl	0	0	0	0	0	0	0	0	7	30	30
Trucks-T	0	0	0	0	1	1	1	1	1	1	0
Tpp-MTC	0	0	0	0	0	0	0	0	0	0	0
NOTANKAGE	17	0	27	27	23	23	23	23	22	27	0
Depots-s-T	13	10	17	10	16	16	16	16	22	22	22
Depots-T	0	1	1	0	1	1	1	1	1	1	1
Trucks-TC	0	0	0	0	1	1	1	1	2	4	1
Rovers-s-T	20	20	20	20	20	20	20	20	20	20	20
Driverlog-T	8	9	14	8	15	15	15	15	19	19	19
Driverlog-s-T	11	14	12	12	10	10	10	10	20	20	20
rovers-T	16	8	16	14	15	15	15	15	13	16	16
PipesWold-MTC	0	1	0	0	0	0	0	0	0	1	1
Zenotravel-T	0	0	0	0	0	0	0	0	0	0	0
Trucks-TC-adl	0	0	0	0	0	0	0	0	0	0	0
Zenotravel-s-T	20	20	20	20	20	20	20	20	20	20	20
Total	238	167	260	229	309	309	301	309	**360**	437	404
IPC-Score	175.1	177.9	183.0	168.4	**277.0**	**277.0**	269.2	**277.0**	242.6	437	249.6

Table 8.14 presents similar results to previous table. Regression models work better than classification ones. LPG-TD has the same results as the unknown domains, which are always better in terms of coverage, but it achieves worse results in terms of score. The results of the LPG-TD are in the column (C % S) because it is the best solver in this test set. The results of these tables show the results of the Virtual Best Solver

(VBS) in both cases which is the upper limit of our technique in terms of score and solved problems. Furthermore, we show the results of a portfolio with the best planners in the training phase. This portfolio is the sequential execution of LPG-td, Yashp2, Yashp3 and TFD. A natural consequence of the portfolio is to solve more problems than every approximation, but the time necessary to find the solution could be high. This situation occurs when the TFD is the solver that finds the solution. This planner starts to find solutions at 1,350 seconds into the execution. It is the reason why the score is not far from for the best planner (LPG-td).

Table 8.14: Performance, in terms of coverage and overall IPC score, of regression and classification EPMs exploited for algorithm selection at know domains.

	Classification				Regression				C & S	VBS	B4P
Domain	All	Sel	NT	T	All	Fea	NT	T			
Openstacks	20	20	18	20	20	17	20	20	20	20	20
Storage	25	29	15	29	23	22	23	22	29	29	28
Satellite-adl	6	9	10	2	11	11	11	11	0	19	11
Openstacks-s	20	20	20	20	18	18	18	18	20	20	20
Openstacks-S	0	0	0	0	20	20	20	20	20	20	20
Satellite_s	8	0	8	7	3	3	3	3	34	34	34
Openstacks-S	0	0	0	0	20	20	20	20	20	20	20
Coverage	79	78	71	78	**115**	111	**115**	114	**143**	162	153
IPC-Score	50.0	57.3	64.2	54.8	**113.1**	110.2	110.2	108.6	88.7	162.0	90.0

8.6.4 Planner Selection Analysis

All the predictive models select one planner per instance (problem), however the selection between the problems in the same domain is sometimes a difficult situation. In this section, we look at the results of the table 8.12 to find out the selection of the planner.

Of the 4 domains introduced in IPC 2014, we noticed that the regression approaches have a better performance on average, so they are able to better generalize in previously unseen domains; the classification approach is unable to select a good planner for the *RTAM* domain, but it makes good decisions in the *MapAnalyser* problems. Finally, we observed that the regression EPMs generally select the same planner for all of the instances within the same domain for this test set (IPC-2014). This is not true for the classification approaches, which usually exploit more planners per domain. Table 8.15 shows the planners selected by the EPMs considered using the different sets of features. Interestingly, in every domain except *Floortile*, the algorithm selection based on regression uses one single planner. This, in combination with the results shown in Table 8.12, supports the hypothesis that a single planner usually performs well on benchmarks from the same domain. Also, by analysing the results shown in Table 8.15, we can extrapolate that the difference in performance between regression EPMs using the selected set of features, and the other sets, mainly arises in the *Driverlog* domain. In that domain, TFD does not solve any problem, thus selecting it has a detrimental effect on performance. It should also be noted that the winner of the IPC-14 temporal track –YAHSP3-MT– is never selected by the regression EPMs, and is selected in only one domain by the classification EPMs. Similarly, the previous version of that planner is rarely used. This is possibly due to the fact that these planners show impressive performance on a very limited number of domains –particularly *RTAM* and *MapAnalyser*, which are not included in the training set–. We also noticed the remarkable performance of the LPG-TD planner; even though it was developed more than a decade ago, it is competitive with the current state-of-the-art of temporal planning. Moreover, it is usually selected by both classification and regression EPMs; thus its performance is good both in training and testing instances, and is possibly "easy" to predict. Finally, Table 8.16 summarises the number of times that each planner was selected by the EPMs built.

Table 8.15: Planners selected using Classification or Regression EPMs, with different sets of features, on IPC 2014 benchamrks.

		Classification				Regression			
		All	Sel	nT	T	All	sel	nT	T
Driver	LPG-TD	0	0	2	0	20	0	20	20
	POPF2	0	3	0	1	0	0	0	0
	TFD	3	14	3	19	0	20	0	0
	Y2	17	3	13	0	0	0	0	0
	ITSAT	0	0	2	0	0	0	0	0
Floor	ITSAT	10	0	20	20	15	20	15	20
	LPG-TD	10	20	0	0	5	0	5	0
Map	LPG-TD	0	0	0	0	20	20	20	20
	POPF2	5	0	0	0	0	0	0	0
	TFD	15	20	14	14	0	0	0	0
	ITSAT	0	0	6	6	0	0	0	0
MatchC.	ITSAT	15	0	9	9	0	0	0	0
	POPF2	0	0	4	4	0	20	0	20
	TFD	5	20	7	7	20	0	20	0
Park.	POPF2	11	0	0	0	0	0	0	0
	Y2	0	20	0	0	0	0	0	0
	Y2-MT	9	0	0	0	20	20	20	20
	Y3-MT	0	0	20	20	0	0	0	0
RTAM	LPG-TD	0	6	0	0	20	20	20	20
	TFD	20	14	17	17	0	0	0	0
	Y3-MT	0	0	3	3	0	0	0	0
Satellite	LPG-TD	0	0	0	0	20	20	20	20
	POPF2	7	20	0	0	0	0	0	0
	TFD	13	0	2	2	0	0	0	0
	ITSAT	0	0	18	18	0	0	0	0
Stor.	LPG-TD	20	20	20	20	20	20	20	20
TMS	ITSAT	20	20	20	20	20	20	20	20
T&O	ITSAT	3	0	0	0	0	0	0	0
	POPF2	6	11	2	2	0	0	0	0
	TFD	11	9	18	18	20	20	20	20

In Figure 8.2, the domains are not sorted alphabetically, but follow the *complexity* of their instances. In this context, the smaller the number of planners that can solve all the problems, the more complex the domain is. According to this intuitive definition, the less complex (easier) domain is Parking, since 6 planners solve all of the problems, and all the planners considered solve at least 6 instances. The two more complex domains are *TMS*, because only one planner is able to solve its benchmark problems, and *TurnAndOpen*, where 3 planners only solve some problems, 10 on average. The complexity of the domains plays a pivotal role in the selection of the algorithm. If a complex domain is included in the training set, it is easier for the EPM to identify the planner(s) correctly to be used in corresponding testing instances. On the other hand, if the domain is not considered in the training set, the capability of the EPM-based approach to select the better planner depends only on the information capabilities of the features and generalisation.

Figure 8.2 provides an overview of the complexity of the testing domains used in IPC 2014, both from the planning and instances perspective. The red line (Solved Problems) represents the proportion of problems solved per domain. A value of 1 indicates that all of the planners are able to solve all of the testing problems; on the other hand, a value of 0 means that no planner can solve any of the testing problems. Similarly, the green line (Planners) sets out the planners' perspective, as the proportion of planners that can solve all the problems of a domain. Figure 8.2 clearly shows that out of the domains considered, 4 are extremely complex for the state-of-the-art domain-independent planners. Intuitively, the complexity of *TMS* and *TurnAndOpen* derives from the fact that their problems need the actions to be executed concurrently, in order to be solved.

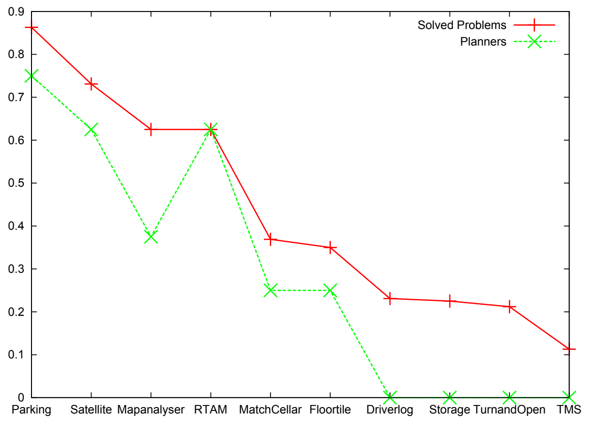


Figure 8.2: The red line (Solved Problems) is the proportion of the problems solved by all the state-of-the-art planners. The green line (Planners) is the proportion of the planners that solved all the problems of the domain.

8.7 Summary

In this chapter, we fill the gap between classical and temporal planning in terms of predictive models. Our work establishes a new extensive set of features that can be extracted from temporal planning problems. In particular, we introduce 71 new temporal-specific features, and merge them with "classical" features that can also be

Table 8.16: Number of times each planner has been selected by classification or regression EPMs exploiting different sets of features. nT and T refer respectively to Non-Temporal and Temporal sets of features.

	Classification				Regression			
	All	Sel	nT	T	All	sel	nT	T
LPG-TD	30	46	22	20	105	80	100	105
YAHSP2	17	23	13	0	0	0	0	0
YAHSP2-MT	9	0	0	0	20	20	20	20
POPF2	29	34	6	7	0	20	20	0
ITSAT	48	20	57	55	35	40	40	35
TFD	67	77	61	77	40	40	20	40
YAHSP3	0	0	0	0	0	0	0	0
Yashp3-MT	0	0	23	23	0	0	0	0

extracted from temporal problems. In total, 139 planning-specific features have been considered for generating both classification and regression EPMs in order to select the algorithm to be executed for solving a target planning task on-line. The analysis carried out in this work: (i) suggests that the performance of many temporal planners can be accurately predicted using EPMs; (ii) gives insights into the reasons that make planners hard to predict, particularly those that run out of memory and those with the concurrency requirements; (iii) provides a valuable and informative set of 11 features that can be used for effectively predicting the performance of the temporal planners; (iv) shows that both temporal-specific and non-temporal features are useful for predicting planners' performance; (v) demonstrates that using EPMs for algorithm selection can improve the current state-of-the-art of temporal planning lightly. Our work also highlights worrying evidence: in terms of coverage, those planners that were introduced more than a decade ago are able to achieve a performance comparable to –and often better– with the most recent planning systems. LPG-TD results have emphasized this idea, in which old problems and domains work better than new planners. This should be a matter of reflection for the planning community. Furthermore, this chapter has set out a great deal of experimentation in temporal domains with the state-of-the-art temporal planners and including EPMs.

We included this chapter after Chapter 7, but this part was developed before.

9 Conclusions and Future Work

In this Chapter, we discuss the conclusions and describe possible future work.

9.1 Conclusions

In this thesis we renew the idea of planning portfolios configured through predictive models. The first hypothesis of this work is that a small group of planners could have a better performance than a large number of them if this small group provides enough diversity. As we proposed in our objectives, we have carried out an experimental study with classical metrics to discover which metric could come up with a subset of planners with enough diversity. We show that this kind of selection might reduce the diversity of the portfolios and we introduce a multi-criteria method to achieve this objective, which has never been used before in AP. The experimental results demonstrate that the simple criterion of a combination of time and quality, without any other information, could create the best static state-of-the-art portfolio.

The second sub-objective of this thesis consists of creating a group of well-informed features that could properly characterize the planning tasks. We propose 50 new features from the SAS$^+$ representation, 10 from the fact balance of the relaxed plan, 14 from the landmark graph and 16 from the heuristic values of the initial state. Furthermore, we incorporate the old features based on PDDL representations to the new set of features. This group of features provides a better characterization for a wide range of planning tasks, even in many cases in which these problems are of equivalent size at the PDDL representation level.

We have created predictive models using the proposed features. These models are trained inductively through a Data Mining process. The set of proposed features partially characterizes the difficulty in the planning problems, which allows us to model the performance of the automated planners. Extracting features is not a disadvantage because the complete set of features is easy to compute, therefore these features are generated in a pre-processing stage, and then used to query predictive models for deciding the set of planners to be run and for how long. Furthermore, we include a feature selection process with a small group of them with comparative results in terms of coverage. These predictive models are proposed as a second method to filter planners and maintain the diversity of the planners by reducing the size of the initial set of planners by about 80% (around 45% from the other planner filtering).

In this thesis, we also propose a group of portfolio configuration strategies based on the previous knowledge. These strategies are created taking into account the different amounts of knowledge acquired previously. On several occasions, a static configuration is enough to configure a portfolio; however, other times it is not. We detail a group of strategies to create static portfolios with uniform time, dynamic portfolios customizable per problem with no time estimation and dynamic portfolios customizable per problem with time estimation. We have completed a study to delimit the number of planners that are necessary to guarantee similar performance to our static approximation. This study shows that five planners are enough to achieve it in the set-up evaluated. These strategies take into account other constraints, such as the time limit, and could be easy adaptable to future requirements (like memory).

The results show that the portfolio configurations using the classification models are able to select a good subset of 5 planners, which with uniformly distributed time outperforms the selection provided randomly and the best planners in terms of quality. Additionally, our experimental evaluation confirms the great performance of IBaCoP and IBaCoP2 in IPC-2014. Configurations using regression models do not have benefits over IBaCoP2. Even though we recognize that estimating the runtime for solving a problem is still very difficult.

These experimental results show that the static strategies have promising results. For instance, our planner filtering (IBaCoP) has similar results to those using predictive models IBaCoP2, the main difference is the number of selected planners in IBaCoP2. With just the 5 planner selection we obtain similar or slightly better scores (quality). Static portfolio configurations (including IBaCoP) are limited by the components and the fixed time limit for each base planner. Their performance has an upper-limit, as computed by MiPlan (Núñez et al., 2015a), which 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 for different base planners or in a more drastic way, all the time for the best planner.

Furthermore, we analyze the diversity of the planner selection methods in the IPC-2014 benchmark to find out the importance of each component and how the models take a different configuration according to the problems. To go further with this process, the confidence analysis on one test domain is included. In this analysis, we show the real differences in the planner's confidence in different problems of a specific domain. The last study demonstrates that our system has a different configuration per domain and per problem.

In this thesis we also verify whether known features for characterizing planning tasks are able to encode knowledge for the classification of hard tasks in scenarios in which performance models have to discriminate between problems with similar characteristics (input configuration). These results show that by considering the most diverse performance data from a set of planners and domains, the performance models behave better systematically than random classifiers. Most of the EPMs trained on homogeneous problem sets are still far from perfect classifiers, so we think there is room for aggregating additional features that characterize other aspects regarding the search space.

However, these results highlight that our proposed features contribute to identifying differences between problems.

Furthermore, we propose a temporal planning portfolio. In this configuration, we include temporal planners and build EPMs to analyze their predictability. This is the first temporal approximation, and the first study of EPMs in this field. Following the hypothesis of this thesis, we have changed the system to adapt and extract the features, including temporal-specific ones and, finally, strategies to build a temporal portfolio per instance. The results show that the portfolio could be useful in different types of planning settings such as temporal, although a static sequential portfolio is enough to improve the current state-of-the-art planners, at least in terms of coverage.

Summarizing, the main contributions of this thesis are:

- To define a scope of the metrics to select a group of potential planners and explain the advantages and disadvantages of using them in Chapter 3.
- To propose a multi-criteria planner selection based on Pareto Efficiency as presented in Chapter 3.
- A new set of features that characterize the planning task in Chapter 4.
- Several overall EPMs based on previous features with a set of 45 domains as detailed in Chapter 4.
- To create several strategies that use EPMs and we propose a new idea to incorporate more than a one planner into a portfolio configuration as we set out in Chapter 5.
- To carry out a study to verify whether some of our features are able to characterize planning tasks even when they have the same structure as explained in Chapter 7.
- The first temporal portfolio as illustrated in Chapter 8.

9.2 Future Work

As future work we want to study the synergy between different automated planners that solve a planning task, incorporating this information for the portfolio configuration. These situations could learn rules such as: if the planner A is good within this planning task then planner B could be better. Furthermore, we want to continue the development of new features in order to improve the results, mainly in the regression task.

Additionally, we want to include other features extracted from search probes (Lipovetzky and Geffner, 2011), without compromising the computational cost of computing them too much. We are also interested in developing random generators of hard problems. The idea consists of making a wrapper of the problem generators available with produces only those instances that are considered above a certain cut-off of difficulty as provided by the corresponding model prediction.

The integration of different planner configurations using algorithm configuration tools, such as SMAC (Hutter et al., 2011) to create automated strategies, is also a promising line of research. This tool constructs explicit regression models to describe the dependence of target algorithm performance on parameter settings. This system has shown to be successful in many areas of AI.

9.3 Publications

Next, the list of publications related to this Thesis is included:

- Isabel Cenamor, Tomás de la Rosa and Fernando Fernández, Mining IPC-2011 Results. In Proceedings of the Third ICAPS Workshop on the International Planning Competition. June, 2012.
- Isabel Cenamor, Tomás de la Rosa and Fernando Fernández, Learning Predictive Models to Configure Planning Portfolios, Proceedings of ICAPS workshop on Planning and Learning. June, 2013.
- Isabel Cenamor, Tomás de la Rosa and Fernando Fernández, IBaCoP and IBaCoP2 Planner. In: Planner description, Deterministic track, International Planning Competition 2014
- Isabel Cenamor, Tomás de la Rosa and Fernando Fernández, LIBaCoP and LIBaCoP2 Planner. In: Planner description, Learning track, International Planning Competition 2014
- Isabel Cenamor, Tomás de la Rosa and Fernando Fernández, The IBaCoP Planning System: Instance-Based Configured Portfolios. Journal of Artificial Intelligence Research (JAIR) No. 56
- Tomás de la Rosa, Isabel Cenamor and Fernando Fernández, Performance modelling of planners from homogeneous problem sets. In the 27th International Conference on Automated Planning and Scheduling (ICAPS-2017).

9.4 Awards

- Winner planner in Sequential Satisficing track in the International Planning Competition 2014
- Runner-up planner in Sequential Multi-core track in the International Planning Competition 2014
-

References

Alcázar, V., Fernández, S., and Daniel, B. (2014). BiFD: Bidirectional fast downward. *IPC 2014 planner abstracts*, pages 18–19. 26

Alcázar, V., Fernández, S., Borrajo, D., and Veloso, M. (2015). Using random sampling trees for automated planning. *AI Communications*, 28(4):665–681. 155

Amadini, R., Gabbrielli, M., and Mauro, J. (2014a). Portfolio approaches for constraint optimization problems. *TPLP*, 8426:21–35. 29

Amadini, R., Gabbrielli, M., and Mauro, J. (2014b). SUNNY: a lazy portfolio approach for constraint solving. *TPLP*, 14(4-5):509–524. 29

Bacchus, F. and Yang, Q. (1994). Downward refinement and the efficiency of hierarchical problem solving. *Artificial Intelligence*, 71(1):43–100. 13

Bäckström, C. and Jonsson, P. (1995). Planning with abstraction hierarchies can be exponentially less efficient. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, pages 1599–1605. Morgan Kaufmann. 10

Bäckström, C. and Nebel, B. (1995). Complexity results for SAS+ planning. *Computational Intelligence*, 11:625–656. 120

Baioletti, M., Chiancone, A., Poggioni, V., and Santucci, V. (2014). Towards a new generation aco-based planner. In *International Conference on Computational Science and Its Applications*, pages 798–807. Springer. 155

Balyo, T. (2014). The freelunch planning system entering ipc 2014. *IPC 2014 planner abstracts*, pages 43 – 44. 26

Biere, A., Heule, M., and van Maaren, H. (2009). *Handbook of satisfiability*, volume 185. IOS press. 20

Blum, A. L. and Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1):245–271. 50

Bonet, B. and Geffner, H. (2000). Planning as heuristic search: New results. In *Recent Advances in AI Planning*, pages 360–372. Springer. 57

Bonet, B. and Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5 – 33. 17

Bonet, B., Loerincs, G., and Geffner, H. (1997). A robust and fast action selection mechanism for planning. In *AAAI/IAAI*, pages 714–719. 57

Bonisoli, A., Gerevini, A. E., Saetti, A., and Serina, I. (2015). Effective plan retrieval in case-based planning for metric-temporal problems. *J. Exp. Theor. Artif. Intell.*, 27(5):603–647. 91

Brafman, R. I. and Domshlak, C. (2003). Structure and complexity in planning with unary operators. *Journal of Artificial Intelligence Research (JAIR)*, 18:315–349. 13

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32. 90, 97

Browne, M. W. (2000). Cross-validation methods. *Journal of mathematical psychology*, 44(1):108–132. 51

Cai, D., Hu, Y., and Yin, M. (2011). SatPlanLM and SatPlanLM-c: Using landmarks and their orderings as constraints. *The 2011 International Planning Competition*, pages 77–78. 156

Cameron, C., Hoos, H. H., and Leyton-Brown, K. (2016). Bias in algorithm portfolio performance evaluation. In Kambhampati, S., editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 712–719. IJCAI/AAAI Press. 21

Celorrio, S. J., Jonsson, A., and Palacios, H. (2015). Temporal planning with required concurrency using classical planning. In Brafman, R. I., Domshlak, C., Haslum, P., and Zilberstein, S., editors, *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015, Jerusalem, Israel, June 7-11, 2015.*, pages 129–137. AAAI Press. 9

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 - ICAPS*. 4

Cenamor, I., de la Rosa, T., and Fernández, F. (2013). Learning predictive models to configure planning portfolios. In *Proceedings of the Workshop on the Planning and Learning - ICAPS*, pages 14–22. 4, 51

Cenamor, I., de la Rosa, T., and Fernández, F. (2014a). Ibacop and ibacop2 planner. *IPC 2014 planner abstracts*, pages 35–38. 90

Cenamor, I., de la Rosa, T., and Fernández, F. (2014b). LIBaCoP and LIBaCoP2 planner. *Eighth International Planning Competition (IPC-8) Planning and Learning Part: planner abstracts*. 27, 91

Cenamor, I., de la Rosa, T., and Fernández, F. (2016). The IBaCoP planning system: Instance-based configured portfolio. *Journal of Artificial Intelligence Research (JAIR)*, 56:657–691. 27

Censor, Y. (1977). Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization*, 4(1):41–59. 3, 39

Coles, A., Coles, A., Fox, M., and Long, D. (2011a). LPRPG: A planner for metric resources. *The 2011 International Planning Competition*, 58. 156

Coles, A., Coles, A., Fox, M., and Long, D. (2011b). popf2: a forward-chaining partial order planner. *The 2011 International Planning Competition*, page 65. 156, 157

Cushing, W. A., Kambhampati, S., Mausam, and Weld, D. S. (2007). When is temporal planning really temporal? In *IJCAI*, pages 1852–1859. 9

de la Rosa, T., Cenamor, I., and Fernández, F. (2017). Performance modelling of planners from homogeneous problem sets. In *ICAPS*. 93

Dietterich, T. G. (2000). Ensemble methods in machine learning. In Kittler, J. and Roli, F., editors, *Multiple Classifier Systems, First International Workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000, Proceedings*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer. 2

Domshlak, C., Hoffmann, J., and Katz, M. (2015). Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, 221:73–114. 82

Dréo, J., Savéant, P., Schoenauer, M., and Vidal, V. (2011). Divide-and-evolve: the marriage of descartes and darwin. *Proceedings of the 7th international planning competition (IPC). Freiburg, Germany*. 91, 155

Eyerich, P., Mattmüller, R., and Röger, G. (2012). Using the context-enhanced additive heuristic for temporal and numeric planning. In *Towards Service Robots for Everyday Environments*, pages 49–64. Springer. 120, 158

Fawcett, C., Helmert, M., Hoos, H., Karpas, E., Röger, G., and Seipp, J. (2011). FD-Autotune: Domain-specific configuration using fast-downward. *Proceedings of the Workshop on the Planning and Learning - ICAPS*, 2011(8). 156

Fawcett, C., Vallati, M., Hutter, F., Hoffmann, J., Hoos, H. H., and Leyton-Brown, K. (2014). Improved features for runtime prediction of domain-independent planners. In *In Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS-14)*. 30, 66

Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208. 12

Fox, M. and Long, D. (2003). PDDL2. 1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)*, 20:61–124. 9, 12

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin. 66

Fuentetaja, R. (2011). The CBP planner. In *International Conference on Automated Planning and Scheduling (ICAPS) Workshop on International Planning Competition (IPC)*. 155

Gerevini, A., Saetti, A., and Serina, I. (2003). Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research (JAIR)*, 20:239–290. 157

Gerevini, A., Saetti, A., and Serina, I. (2006). An approach to temporal planning and scheduling in domains with predictable exogenous events. *Journal of Artificial Intelligence Research (JAIR)*, 25:187–231. 9, 34, 157

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)*. 4, 25

Gerevini, A., Saetti, A., and Vallati, M. (2014). Planning through automatic portfolio configuration: The PbP approach. *Journal of Artificial Intelligence Research (JAIR)*, 50:639–696. 4, 25, 86

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL — the planning domain definition language version 1.2. Technical report, Yale Center for Computational Vision and Control. 10, 12, 15

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated planning: theory & practice*. Elsevier. 1, 2, 7

Gomes, C. P. and Selman, B. (2001). Algorithm portfolios. *Artificial Intelligence*, 126(1):43–62. 4

Grabczewski, K. and Jankowski, N. (2005). Feature selection with decision tree criterion. In *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, pages 212–217. IEEE. 64

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18. 124

Hall, M. A. (1999). *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato. 98

Hall, M. A. (2000). Correlation-based feature selection for discrete and numeric class machine learning. In Langley, P., editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 359–366. Morgan Kaufmann. 98

Han, J., Kamber, M., and Pei, J. (2011). *Data mining: concepts and techniques*. Elsevier. 50

Hanley, J. A. and McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36. 97

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284. 125

Helmert, M. (2004). A planning heuristic based on causal graph analysis. In Zilberstein, S., Koehler, J., and Koenig, S., editors, *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 161–170. AAAI. 12, 13, 57

Helmert, M. (2006). The Fast Downward planning system. *Journal of Artificial Intelligence Research (JAIR)*, 26:191–246. 4, 18, 24, 157

Helmert, M. (2009). Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence*, 173:503–535. 10, 12, 18, 53

Helmert, M. and 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)*. 19, 57

Helmert, M. and 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)*, pages 140–147. 57

Helmert, M., Röger, G., Seipp, J., Karpas, E., Hoffmann, J., Keyder, E., Nissim, R., Richter, S., and Westphal, M. (2011). Fast downward stone soup. *The Seventh International Planning Competition*, IPC-7 planner abstracts:38. 24, 32, 86, 155, 157

Hoffmann, J. (2003). The metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research (JAIR)*, 20:291–341. 62

Hoffmann, J. (2011). Analyzing search topology without running any search: On the connection between causal graphs and h+. *Journal of Artificial Intelligence Research (JAIR)*, 41:155–229. 29, 30, 123

Hoffmann, J., Edelkamp, S., Thiébaux, S., Englert, R., Liporace, F., and Trüg, S. (2006). Engineering benchmarks for planning: the domains used in the deterministic part of ipc-4. *Journal of Artificial Intelligence Research (JAIR)*, 26:453–541. 83

Hoffmann, J. and Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research (JAIR)*, 14:253–302. 17, 29, 57, 58

Hoffmann, J., Porteous, J., and Sebastia, L. (2004). Ordered landmarks in planning. *Journal of Artificial Intelligence Research(JAIR)*, 22:215–278. 19

Hoos, H., Kaminski, R., Schaub, T., and Schneider, M. T. (2012). aspeed: ASP-based solver scheduling. *ICLP (Technical Communications)*, 17:176–187. 29

Howe, A. and Dahlman, E. (2002). A critical assessment of benchmark comparison in planning. *Journal of Artificial Intelligence Research (JAIR)*, 17:1 – 33. 123

Howe, A. E., Dahlman, E., Hansen, C., Scheetz, M., and von Mayrhauser, A. (1999). Exploiting competitive planner performance. In Biundo, S. and Fox, M., editors, *Recent Advances in AI Planning, 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999, Proceedings*, volume 1809 of *Lecture Notes in Computer Science*, pages 62–72. Springer. 3, 24

Howey, R., Long, D., and Fox, M. (2004). Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 294–301. IEEE Computer Society. 15

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer. 25, 141

Hutter, F., Hoos, H. H., Leyton-Brown, K., and Stützle, T. (2009). ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research (JAIR)*, 36:267–306. 25

Hutter, F., Xu, L., Hoos, H., and Leyton-Brown, K. (2015). Algorithm runtime prediction: Methods and evaluation (extended abstract). In Yang, Q. and Wooldridge, M., editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 4197–4201. AAAI Press. 29, 66

Huzak, M. (2011). Chi-square distribution. *International Encyclopedia of Statistical Science*, pages 245–246. 99

John, G. H. and Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc. 97

Jonsson, P. and Bäckström, C. (1998). State-variable planning under structural restrictions: Algorithms and complexity. *Artificial Intelligence Journal*, 100(1-2):125–176. 13

Karegowda, A. G., Manjunath, A., and Jayaram, M. (2010). Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management*, 2(2):271–277. 97

Katz, M. and Domshlak, C. (2011). Planning with implicit abstraction heuristics. *7th International Planning Competition (IPC)*, pages 46–49. 156

Katz, M. and Hoffmann, J. (2013). Red-black relaxed plan heuristics reloaded. In *Proceedings of the Sixth Annual Symposium on Combinatorial Search, SOCS 2013, Leavenworth, Washington, USA, July 11-13, 2013.* 57

Katz, M., Hoffmann, J., and Domshlak, C. (2013). Red-black relaxed plan heuristics. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA.* 57

Kautz, H. A., Selman, B., and Hoffmann, J. (2006). SatPlan: Planning as satisfiability. In *Abstracts of the 5th International Planning Competition.* 20, 49, 156

Kavuluri, B. R. (2011). Extending temporal planning for the interval class. *7th International Planning Competition (IPC)*, pages 79–82. 156

Keyder, E., Richter, S., and Helmert, M. (2010). Sound and complete landmarks for and/or graphs. In *ECAI*, pages 335–340. 18, 19

Knoblock, C. A. (1994). Automatically generating abstractions for planning. *Artificial intelligence*, 68(2):243–302. 13

Kohavi, R. (1995). The power of decision tables. In *Machine Learning: ECML-95*, pages 174–189. Springer. 66, 128

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324. 98

Linares López, C., Celorrio, S. J., and Olaya, A. G. (2015). The deterministic part of the seventh international planning competition. *Artificial Intelligence*, 223:82–119. 39, 43

Lindauer, M. T., Hoos, H. H., and Hutter, F. (2015a). From sequential algorithm selection to parallel portfolio selection. In Dhaenens, C., Jourdan, L., and Marmion, M., editors, *Learning and Intelligent Optimization - 9th International Conference, LION 9, Lille, France, January 12-15, 2015. Revised Selected Papers*, volume 8994 of *Lecture Notes in Computer Science*, pages 1–16. Springer. 29

Lindauer, M. T., Hoos, H. H., Hutter, F., and Schaub, T. (2015b). Autofolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence Research (JAIR)*, 53:745–778. 29, 71

Lipovetzky, N. and Geffner, H. (2011). Searching for plans with carefully designed probes. In *In Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS-11)*, pages 154–161. 141, 157

López, C. L., Celorrio, S. J., and Helmert, M. (2013). Automating the evaluation of planning systems. *AI Commun.*, 26(4):331–354. 15

Lu, Q., Xu, Y., Huang, R., and Chen, Y. (2011). The roamer planner random-walk assisted best-first search. *The 2011 International Planning Competition*, pages 73–76. 156

Malitsky, Y., Sabharwal, A., Samulowitz, H., and Sellmann, M. (2013). Algorithm portfolios based on cost-sensitive hierarchical clustering. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 608–614. AAAI Press. 2

Malitsky, Y., Wang, D., and Karpas, E. (2014). The allpaca planner: All planners automatic choice algorithm. *IPC 2014 planner abstracts*, pages 71–73. 26, 71, 79

Nakhost, H. and Müller, M. (2010). Action elimination and plan neighborhood graph search: Two algorithms for plan improvement. In *icaps*, pages 121–128. 82

Nakhost, H., Müller, M., Valenzano, R., and Xie, F. (2011). Arvand: the art of random walks. *The Seventh International Planning Competition*, IPC-7 planner abstracts:15–16. 156

Nebel, B. (2000). On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research (JAIR)*, 12:271–315. 83

Núñez, S., Cenamor, I., and Virseda, J. (2014). NuCeLaR. *IPC 2014 planner abstracts*, pages 58–61. 26

Núñez, S., Borrajo, D., and Linares López, C. (2015a). Automatic construction of optimal static sequential portfolios for AI planning and beyond. *Artificial Intelligence*, 226:75–101. 25, 32, 140

Núñez, S., Borrajo, D., and Linares López, C. (2015b). Sorting sequential portfolios in automated planning. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1638–1644. 70

Núñez, S., Borrajo, D., and López, C. L. (2014). Miplan and dpmplan. *IPC 2014 planner abstracts*, pages 13–16. 26

Olsen, A. and Bryce, D. (2011). Randward and lamar: Randomizing the FF heuristic. *The 2011 International Planning Competition*, page 55. 90, 156, 157

O'Mahony, M. (1986). *Sensory evaluation of food: statistical methods and procedures*, volume 16. CRC Press. 67

Porteous, J., Sebastia, L., and Hoffmann, J. (2001). On the extraction, ordering, and usage of landmarks in planning. In *Sixth European Conference on Planning*. Citeseer. 18

Quinlan, J. R. (1993). *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann. 64, 65, 97, 124, 125

Rankooh, M. F., Mahjoob, A., and Ghassem-Sani, G. (2012). Using satisfiability for non-optimal temporal planning. In *Logics in Artificial Intelligence*, pages 176–188. Springer. 120, 123, 158

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, 15:65 – 118. 117

Richter, S., Helmert, M., and Westphal, M. (2008). Landmarks revisited. In Fox, D. and Gomes, C. P., editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 975–982. AAAI Press. 19, 57

Richter, S. and Westphal, M. (2010). The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research (JAIR)*, 39(1):127–177. 18, 157

Richter, S., Westphal, M., and Helmert, M. (2011). Lama 2008 and 2011. *The Seventh International Planning Competition*, IPC-7 planner abstracts:50. 157

Rintanen, J. (2011). Madagascar: Efficient planning with SAT. *The Seventh International Planning Competition*, IPC-7 planner abstracts:61. 157

Rintanen, J. (2014). Madagascar: Scalable planning with SAT. *Proceedings of the 8th International Planning Competition (IPC-2014)*. 21

Rintanen, J. (2015). Impact of modeling languages on the theory and practice in planning research. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 10

Rintanen, J., Heljanko, K., and Niemelä, I. (2006). Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, 170(12):1031–1080. 21

Rizzini, M., Fawcett, C., Vallati, M., Gerevini, A. E., and Hoos, H. H. (2015). Portfolio methods for optimal planning: an empirical analysis. In *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*, pages 494–501. IEEE. 30

Roberts, M. and Howe, A. (2009). Learning from planner performance. *Artificial Intelligence*, 173:536–561. 3, 29, 49, 66

Roberts, M., Howe, A. E., Wilson, B., and desJardins, M. (2008). What makes planners predictable? In *In Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS-08)*, pages 288–295. 3, 29, 119

Rodriguez, J. J., Kuncheva, L. I., and Alonso, C. J. (2006). Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630. 66, 97, 125

Russell, S. and Norvig, P. (1995). Artificial intelligence: a modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*. 1, 3

Sadraei, R. and Agmadi, A. (2014). USE: The useful operator selection. *IPC 2014 planner abstracts*, pages 71 – 73. 26

Schwefel, H.-P., Wegener, I., and Weinert, K. (2013). *Advances in computational intelligence: Theory and practice*. Springer Science & Business Media. 85

Seipp, J., Braun, M., Garimort, J., and Helmert, M. (2012). Learning portfolios of automatically tuned planners. In McCluskey, L., Williams, B., Silva, J. R., and Bonet, B., editors, *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012*. AAAI. 25, 70

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

Sussman, G. J. (1975). *A computer model of skill acquisition*, volume 1. American Elsevier Publishing Company New York. 4

Valenzano, R., Nakhost, H., Müller, M., Schaeffer, J., and Sturtevant, N. (2014). Arvandherd 2014. *IPC 2014 planner abstracts*, pages 11 – 14. 26

Vallati, M. (2012). A guide to portfolio-based planning. In *Multi-disciplinary Trends in Artificial Intelligence*, pages 57–68. Springer. 3, 22

Vallati, M., Chrpa, L., and Kitchin, D. (2014). ASAP: an automatic algorithm selection approach for planning. *International Journal on Artificial Intelligence Tools*, 23(06):1460032. 26, 27

Vallati, M., Chrpa, L., and Kitchin, D. E. (2015). Portfolio-based planning: State of the art, common practice and open challenges. *AI Communications*, 29:1–17. 3

Vidal, V. (2004). A lookahead strategy for heuristic search planning. In *In Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, pages 150–160. 58

Vidal, V. (2011). Yahsp2: Keep it simple, stupid. *The 2011 International Planning Competition*, pages 83–90. 157

Vidal, V. (2014). YAHSP3 and YAHSP3-MT in the 8th international planning competition. In *Proceedings of the 8th International Planning Competition*. 158

Vidal, V. and Geffner, H. (2006). Branching and pruning: An optimal temporal POCL planner based on constraint programming. *Artificial Intelligence*, 170(3):298–335. 155

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd Edition, Morgan Kaufmann. 3, 50, 64, 66

Xie, F., Müller, M., and Holte, R. (2014a). Adding local exploration to greedy best-first search in satisficing planning. In *AAAI*, pages 2388–2394. 82

Xie, F., Müller, M., and Holte, R. (2014b). Jasper: the art of exploration in greedy best first search. In *Planner abstracts*. IPC-2014. 82

Xie, F., Valenzano, R. A., and Müller, M. (2013). Better time constrained search via randomization and postprocessing. In *ICAPS*. 82

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)*, pages 210–216. 2

Xu, L., Hutter, F., Hoos, H., and Leyton-Brown, K. (2012a). Evaluating component solver contributions to portfolio-based algorithm selectors. In *Theory and Applications of Satisfiability Testing–SAT 2012*, pages 228–241. Springer. 29, 71

Xu, L., Hutter, F., Hoos, H., and Leyton-Brown, K. (2012b). Features for sat. *University of British Columbia,, Tech. Rep.* 30

Xu, L., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2008). Satzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research (JAIR)*, 32:565–606. 2, 30, 49, 120

Zhu, L. and Givan, R. (2003). Landmark extraction via planning graph propagation. *ICAPS Doctoral Consortium*, pages 156–160. 19

Appendix A

Portfolio Base Planners

This annexe presents a description of all planners included in this thesis. Section A.1 explains a set of initial ones that were not included in the final version of the portfolio. Section A.2 explains the planners that are included in the planner filtering and Section A.3 explains temporal planners.

A.1 Initial Planners from IPC-2011

The following list is the set of planners in the pre-selection in the initial state which are not included in Section A.

- ACOPlan & ACOPlan2 (Baioletti et al., 2014) are planners based on the ant colony optimization framework, in which a colony of planning ants searches for near optimal solution plans with respect to an overall plan cost metric.
- BRT: Biased Rapidly-exploring Tree (Alcázar et al., 2015) is a planner based on a rapidly exploring random tree adapted to the FD system. This planner estimates which propositions are more likely to be achieved throughout some solution plan and uses that estimation in order to sample more relevant intermediates states. These samples are computed using the planning graph with landmarks as support.
- CBP & CBP2 (Fuentetaja, 2011): Cost-Based Planner is a planner that carries out a heuristic search with a numerical heuristic and a selection of actions extracted from a relaxed planning graph.
- CPT4 (Vidal and Geffner, 2006) is a planner based on constraint programming that integrates existing lower limits with novel representations and propagation rules that manage to prune the search space considerably.
- Dae-YAHSP (Dréo et al., 2011) is a planner based on Evolutionary Computation. It is a Divide-and-Conquer strategy driven by an evolutionary algorithm.
- Fast Downward Stone Soup-1 (Helmert et al., 2011) is anoother combination of heuristics and search algorithms like A.2.

- ForkUniform (Katz and Domshlak, 2011) is a sequential planner that iteratively invoke weighted $A^\star$ heuristic search. This search process uses admissible implicit abstraction heuristics.
- LPRPG (Coles et al., 2011a) is a planner that is designed to solve problems using metric resources according to linear constraints.
- Madagascar-p is a parallel version of Madagascar
- POPF2 (Coles et al., 2011b) is a planner that uses forward-chaining search, expanding a partial-order rather than the conventional total-order: steps added to the plan are ordered after a subset of those in the plan.
- Randward (Olsen and Bryce, 2011) is a modification of the LAMA planner that includes a randomized FF heuristic.
- Roamer (Lu et al., 2011) is a planner that uses a random-walk assisted best-first search algorithm for planning, which invokes a random walk procedure to find exits when the best-first search is stuck on a plateau[1].
- Sharaabi (Kavuluri, 2011) is a planner which tackles required concurrency with overall preconditions requiring continuous support.
- SatPlanLM-c (Cai et al., 2011) is a planner that built on top of SatPlan (Kautz et al., 2006). This planner compiles the planning task into several SAT tasks using the concept of parallel steps and supports action costs.
- YAHSP2 is a non-parallel version of YAHSP2-MT

A.2 Planner Filtering Planners

The following list is the set of planners pre-selected as candidates from the Pareto-dominance filtering described in Section 3.3.

- Arvand (Nakhost et al., 2011): is a stochastic planner that uses Monte Carlo random walks to balance exploration and exploitation in a 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 et al., 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.

[1] A well-observed phenomenon that explores a large number of states without reducing the heuristic function value

- 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 and Westphal, 2010, Richter et al., 2011) is a propositional planner based on the combination of landmark count heuristic and the FF heuristic. The search carries out a set of weighted A^* with iteratively decreasing weights. The planner was developed within the FD Planning System (Helmert, 2006).

- Lamar (Olsen and 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 and Geffner, 2011): exploits the idea of wisely constructed lookaheads or *probes*, which are action sequences computed without searching for 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 a stochastic local search in the space of particular action graphs derived from the planning problem specification.

A.3 Temporal Planners

The following list the set of planners selected as candidates from temporal approximation at Section 8.1.

- LPG (Gerevini et al., 2003) is a planner using local search for solving planning graphs.

- POPF2 (Coles et al., 2011b) is a Forward-Chaining Partial Order Planner that uses forward-chaining search, expanding a partial-order rather than the conventional total-order.

- Yahsp2 and Yahsp2-MT (Vidal, 2011) are planners that compute a lookahead from relaxed plans and use them in the forward state-space heuristic search.

- Temporal Fast Downward (TFD) (Eyerich et al., 2012) is based on the Fast Downward planning system and uses an adaptation of the context-enhanced additive heuristic to guide the search in the temporal state space induced by the given planning problem.
- ITSAT (Rankooh et al., 2012) is an SAT-based (satisfiability based) temporal planner capable of temporally expressive planning.
- Yahsp3 and Yahsp3-MT (Vidal, 2014) are forward state-space heuristic search planners that embed a lookahead policy based on an analysis of relaxed plans. They are the updated versions of Yahsp2 and Yahsp2-MT.

Appendix B

Methods of Empirical Evaluation

In this thesis, there are other metrics that are taken into account in the learning phase. This metrics, used to evaluate the predictive models, will be created in the first part of this thesis. In the following paragraphs we introduce several high-level concepts in evaluating machine learning models: evaluation metrics, test data sets and mechanisms, starting with metrics for classification.

There are many ways of measuring classification performance. Accuracy, confusion matrix, log-loss, Area Under the Curve (AUC) and precision-recall are some of them. I shall explain the important metrics for this thesis and other ones that are related.

- **Accuracy**: measures how often the classifier makes the correct prediction. It is the ratio between the number of correct predictions and the total number of predictions (the number of test data points). $ACC = (\ True\ Positive + True\ negative)/(Total\ Population)$

- **Confusion matrix**: (or confusion table) shows a more detailed breakdown of correct and incorrect classifications for each class. The rows of the matrix correspond to ground truth labels, and the columns represent the prediction. The table B.1 is an example of a confusion matrix.

Table B.1: Confusion Matrix

	Predicted as positive	Predicted as negative
Labeled as positive	True Positive	False Positive
Labeled as negative	False Negative	True negative

- **AUC (ROC area)**: is the area under the plot that shows the sensitivity of the classifier by plotting the rate of true positives (TPR) to the rate of false positives (FPR). In other words, it shows you how many correct positive classifications can be gained as you allow for more and more false positives. The perfect classifier

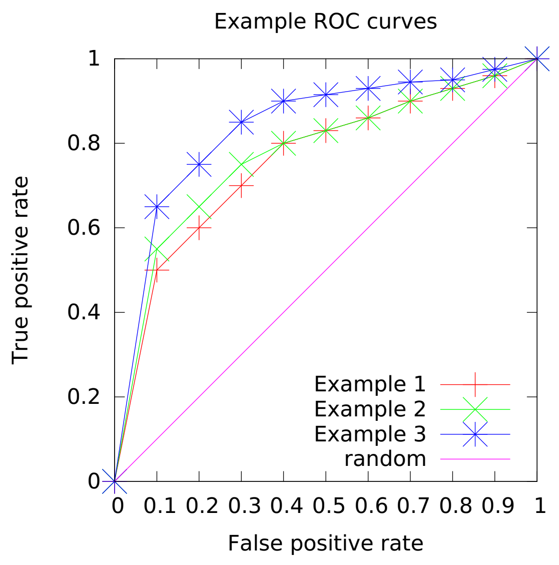


Figure B.1: Example of several ROC curves in contracts with a random ROC area.

that makes no mistakes would hit a true positive rate of 100% immediately, without incurring any false positives. Where the TPR and FPR follow the formula B.1 and B.2. The figure B.1 shows an example of the comparison between 4 different ROC curves.

$$TPR = True\ Positive/(True\ Positive + False\ Negative) \tag{B.1}$$

$$FPR = False\ Positive/(False\ positive + True\ negative) \tag{B.2}$$

- **Relative Absolute Error**: is a measure of the uncertainty of the measurement compared to the size of the measurement. The formula B.3 is made up of the true value Θ, the value estimated using algorithm such as $\widehat{\Theta}$ and the mean value is $\overline{\Theta}$.

$$RAE = \frac{\sum_{n=1}^{N} |\widehat{\Theta}_i - \Theta_i|}{\sum_{n=1}^{N} |\overline{\Theta}_i - \Theta_i|} \tag{B.3}$$

There is another term known as the measure of dispersion which indicates the extent to which the observations are dispersed or spread around the center. The most common measures of dispersion are standard deviation and variance.

When the dispersion is less, the values would be close together or thus close to the center. On the other hand, when the dispersion is more, the values would be far apart and farther from the center.

- **Standard Deviation**: is a measure that is used to quantify the amount of variation or dispersion in a set of data values (Formula B.4).

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{1}^{N}(\theta_i - \bar{\theta})^2} \tag{B.4}$$

- **Index of Dispersion** (also known as VMR i.e. variance-to-mean ratio) is the proportion between the variance squared and the average. This index of dispersion which represents a summary statistic indicates the magnitude of the dispersion. It is the measure of how far the observed frequencies follow the Poisson distribution.

Table B.2: Interpretation of index of Dispersion

VMR	Interpretation	Distribution
0	Not dispersed	Constant random variable
0 to 1	Under dispersed	Binomial distribution
> 1	Over dispersed	Negative binomial distribution
1	–	Poisson distribution

Table B.2 shows that aa interpretation of dispersion is as follows, when the $VMR = 1$, the distribution is a Poisson. If $0 < VMR < 1$ the distribution is under dispersed, binomial distributions have less than 1. If $VMR > 1$ the distribution is over dispersed, negative binomial distributions have s WMR greater than 1. The last case, when $VMR = 0$ there is no dispersion in the data.

- **Coefficient of variation**(or relative standard deviation (RSD)) is a is a standardized measure of dispersion of a probability distribution or frequency distribution. It is the ratio between the standard deviation and the mean. $C_v = \sigma/\mu$

Appendix C

Appendix: Training Results

In this section appear the results of the candidate planners in the training phase. In each table, the last column is the number of problems included for training. The time for this experimentation is $247,690,800$ seconds.

Table C.1: Solved problems in the training phase IPC-2005

	Lama11	Probe	FDA1	Lama08	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

Table C.2: Solved problems in the training phase IPC-2008.

	Lama11	Probe	FDA1	Lama08	FDA2	Lamar	Arvand	FDSS2	ya2-mt	LPG	M	#
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
opens.-adl	31	31	31	31	31	31	31	31	0	1	16	31
opens.	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

Table C.3: Solved problems in the training phase IPC-2011.

	Lama11	Probe	FDA1	Lama08	FDA2	Lamar	Arvand	FDSS2	ya2-mt	LPG	M	#
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

Table C.4: Solved problems in the training phase. IPC-2008 in the learning track.

	Lama11	Probe	FDA1	Lama08	FDA2	Lamar	Arvand	FDSS2	ya2-mt	LPG	M	#
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

Table C.5: Solved problems in the training phase. IPC-2011 in the learning track.

	Lama11	Probe	FDA1	Lama08	FDA2	Lamar	Arvand	FDSS2	ya2-mt	LPG	M	#
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

Appendix D

Appendix: International Planning Competition

The results of the International Planning Competition 2014 appear in this section for all the track we participated in (sequential satisficing, agile, multi-core and learning). Our approximation appears in blue in the tables. These results are the raw results that appeared at ICAPS-2014. Moreover, some bugs arose during the execution of IPC-2014, as some issues in the domain models required updates [1], and some planners were updated [2]. These issues were also fixed prior to running the experimental evaluation presented in this thesis. That the reason why both results does not complexity agree.

Hardware Platform at Sequential Satisficing: The competition run in a cluster of 256 cores. Each core is an AMD Processor 2.39 Ghz. Up to 4GB of RAM and 200 GB of hard disk memory will be available for each planner. Each planner will be run in a single node and no planner is allowed to run in more than one simultaneously (although in the multicore track it is allowed to use all the 4 cores of a node). No GPU is available. Memory and time will be externally limited.

The learning track has 6 domains chosen from previous competitions: elevators, floortile, nomystery, parking, spanner, and transport. At the start of a six-week learning stage, competitors were provided with generators for these domains, a representative set of training problems, and guidelines for the evaluation distributions. After the learning stage was complete, competitors were provided with runs from selected training problems to ensure that their planner was performing as expected. Problems found in these runs were corrected before gathering the final results. For the final evaluation, 5 problems from each domain were randomly generated from the distributions, resulting in 30 instances of problems. The planners were run on the EC2 cloud computer platform with the support of a generous grant from Amazon Web Services; each computer platform had a computer equivalent of 2 cores and 3.75 GB memory and ran Ubuntu 12.04 LTS. To account for variations in the actual computing resources on the cloud platform, each planning system was run 30 times with and without domain knowledge on each instance of a problem.

[1]https://helios.hud.ac.uk/scommv/IPC-14/benchmark.html,Accessed:2015-07-29
[2]https://helios.hud.ac.uk/scommv/IPC-14/errPlan.html,Accessed: 2015-07-29

Table D.1: Results in terms of quality of all competitor planners in 7 domains of the competition (sequential Satisficing track).

	Tetris	Barman	Cave	Childsnack	Citycar	Floortile	Hiking
IBaCoP2	4.02	16.38	7.00	14.98	6.95	18.23	18.04
IBaCoP	6.28	16.27	7.00	15.29	7.32	15.18	18.65
mercury	14.38	13.94	3.00	0.00	3.99	2.00	16.46
MIPlan	7.46	16.54	7.00	18.22	4.69	4.10	18.14
jasper	9.52	19.78	8.00	0.00	8.89	2.00	17.19
uniform	11.52	17.83	7.00	1.23	12.74	1.53	17.01
cedalion	3.20	16.85	7.00	0.71	7.71	7.98	18.74
arvandherd	14.63	18.12	7.00	5.57	19.39	2.00	19.00
fdss-2014	9.87	11.64	7.00	1.36	5.00	2.00	17.44
dpmplan	1.80	16.57	7.00	18.46	5.82	1.97	16.28
use	3.76	15.12	0.00	0.00	1.65	9.17	8.79
nucelar	6.60	15.89	7.00	3.66	7.49	3.37	17.24
rpt	9.41	0.00	3.00	2.98	3.11	19.30	17.35
bfs-f	11.03	8.89	7.94	0.00	0.00	9.71	2.00
bifd	8.67	0.62	3.00	1.16	3.16	19.30	15.25
dae_yahsp	0.00	0.35	0.00	9.36	0.00	0.46	8.12
freelunch	3.74	0.00	3.00	17.43	0.00	7.52	1.91
yahsp3-mt	0.67	6.60	0.00	0.00	0.00	1.22	3.81
yahsp3	0.54	1.33	0.00	0.00	0.00	1.09	4.80
planets	0.00	0.00	5.97	0.00	3.96	1.00	0.77

Table D.2: Results in terms of quality of all competitor planners in 7 domains of the competition and the overall score (sequential Satisficing track).

	Mainte.	Openstacks	Parking	Thoughtful	Transport	Visitall	GED	**Total**
IBaCoP2	16.71	5.15	5.28	15.75	7.01	13.32	17.40	162.20
IBaCoP	16.81	3.58	1.74	13.76	9.94	14.18	16.73	156.45
mercury	5.06	19.69	15.64	0.00	20.00	19.88	19.01	138.66
MIPlan	16.62	9.07	11.08	11.18	0.00	8.18	17.72	142.55
jasper	9.28	17.28	12.91	0.00	7.59	15.18	17.27	135.37
uniform	9.36	10.83	9.74	0.00	9.25	19.56	15.65	131.74
cedalion	14.15	17.08	4.29	0.00	5.14	19.49	15.01	134.14
arvandherd	13.17	14.22	0.69	0.00	4.53	0.68	18.09	122.47
fdss-2014	16.63	17.63	10.86	0.00	4.06	8.78	15.61	118.03
dpmplan	15.07	10.15	0.00	13.18	0.00	3.41	15.79	123.70
use	15.03	15.76	3.65	0.00	6.27	14.80	13.15	103.38
nucelar	16.63	2.73	1.41	0.00	0.00	3.38	16.02	94.84
rpt	10.82	10.61	4.70	0.00	3.57	0.06	13.33	88.85
bfs-f	7.72	0.00	19.94	5.00	3.57	18.99	1.33	85.08
bifd	7.83	5.76	5.38	0.00	3.52	0.00	13.33	78.32
dae_yahsp	0.00	0.00	0.00	17.10	9.02	17.26	2.52	64.19
freelunch	15.26	9.82	0.00	0.00	0.02	2.51	0.00	57.48
yahsp3-mt	0.00	0.00	1.37	14.42	10.74	10.73	8.94	57.83
yahsp3	0.00	0.00	0.00	6.93	8.03	17.08	8.31	47.57
planets	0.00	0.00	0.00	13.25	0.00	0.00	0.00	24.95

Table D.3: Results in terms of quality of all competitor planners in 7 domains of the competition (sequential Agile track).

	Tetris	Barman	Cave	Childsnack	Citycar	Floortile	Hiking
yahsp3	0.44	0.45	0.00	0.00	0.00	0.85	11.29
mpc	2.96	0.31	1.97	7.00	7.17	20.00	4.16
m	0.00	0.00	4.35	17.10	5.46	20.00	1.84
probe	0.00	18.07	0.53	0.00	7.88	1.30	9.55
bfs-f	0.40	15.37	6.70	6.00	2.96	1.67	0.75
cedalion	2.33	11.96	2.46	0.00	2.62	0.75	2.93
freelunch	20.00	0.00	0.00	0.93	0.00	0.70	2.12
yahsp3-mt	0.92	0.00	0.00	0.61	0.00	0.00	14.00
arvandherd	4.68	6.98	2.21	2.57	12.72	0.61	7.98
IBaCoP	1.31	0.40	2.79	9.47	2.55	4.89	5.15
use	0.97	12.14	0.00	3.96	0.00	2.33	3.53
jasper	0.97	10.35	3.96	0.00	2.38	1.00	4.28
mercury	1.03	4.28	0.98	1.61	0.84	1.13	3.29
siw	0.00	0.00	0.00	0.00	2.05	0.00	1.76
IBaCoP2	0.00	0.39	2.46	6.96	1.24	4.38	4.71

Table D.4: Results in terms of quality of all competitor planners in 7 domains of the competition and the overall score (sequential Agile track).

	Mainte.	Openstacks	Parking	Thoughtful	Transport	Visitall	GED	**Total**
yahsp3	0.00	0.00	0.00	9.02	20.00	19.58	20.00	61.62
mpc	12.34	0.00	5.66	3.49	0.00	0.00	4.90	65.06
m	16.31	0.00	0.00	2.56	0.00	0.00	0.00	67.63
probe	6.41	0.00	0.00	13.02	1.74	1.31	6.88	59.82
bfs-f	3.69	0.00	2.22	15.16	1.00	2.33	4.61	58.26
cedalion	5.34	3.72	1.30	0.00	1.97	15.95	10.71	51.32
freelunch	11.04	20.00	0.00	0.00	2.76	2.57	0.00	60.12
yahsp3-mt	0.00	0.00	0.00	16.76	15.97	4.52	0.50	52.78
arvandherd	5.65	2.50	0.00	0.00	1.29	1.25	4.73	48.44
IBaCoP	7.06	0.00	0.00	7.12	2.64	1.93	5.37	45.32
use	8.24	2.50	0.00	0.00	0.98	2.91	9.81	37.55
jasper	4.43	2.51	0.00	0.00	0.96	1.60	9.85	32.43
mercury	5.66	2.55	0.00	0.00	2.04	3.53	9.79	26.94
siw	0.00	0.00	12.00	17.29	0.32	1.58	0.76	35.01
IBaCoP2	6.17	0.00	0.00	1.46	0.00	1.43	5.13	29.19

Table D.5: Results in terms of quality of all competitor planners in 7 domains of the competition (sequential multi-core track)

	Barman	Cave	Childsnack	Citycar	Floortile	GED	Hiking
arvandherd	20.00	7.00	5.50	19.73	2.00	19.13	20.00
IBaCoP	0.00	6.30	8.69	8.78	13.82	18.78	0.00
use	16.04	0.00	5.79	1.00	1.93	15.58	10.11
IBaCoP2	0.00	6.30	9.00	8.38	11.69	18.53	0.00
nucelar	0.00	3.96	1.18	4.26	3.50	17.57	0.00
miplan	0.00	4.95	19.00	4.22	2.69	0.00	0.00
yahsp3-mt	6.95	0.00	4.63	0.00	1.20	10.16	4.69
dae_yahsp	0.58	0.00	0.00	0.00	0.00	0.22	0.64
planets	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table D.6: Results in terms of quality of all competitor planners in 7 domains of the competition and the overall score (sequential Multi-core track).

	Mainte.	Openstacks	Parking	Tetris	Thoughtful	Transport	Visitall	**Total**
arvandherd	15.19	17.46	1.00	17.11	0.00	5.77	3.41	149.90
ibacop	15.89	13.54	0.73	0.00	11.74	3.76	19.89	102.03
use	12.00	14.22	8.31	2.77	0.00	6.77	14.04	94.51
ibacop2	16.92	0.00	0.00	0.00	11.80	7.66	0.00	90.29
nucelar	16.47	1.84	1.80	11.39	8.00	2.52	10.92	72.48
miplan	16.78	4.00	5.00	2.32	10.00	4.33	9.85	73.28
yahsp3-mt	0.00	0.00	2.31	0.78	14.24	17.71	11.83	62.67
dae_yahsp	0.00	0.00	0.00	0.00	3.61	0.57	1.89	5.61
planets	0.00	0.00	0.00	0.00	2.88	0.00	0.00	2.88

Table D.7: Planning & Learning track IPC-2014. The first column is the ranking results, the column third and fourth are results in terms of quality, and the last ones are in terms of coverage.

ranking	Planner	Quality		Coverage	
		Nknowledge	Knowledge	Nknowledge	Knowledge
8	eroller	1.06	12.51	90	540
9	rollent	1.06	11.83	90	600
5	agap	7.98	15.94	237	489
2	cedalion	8.12	19.98	300	603
3	smac	8.12	17.45	300	749
10	badff	11.44	10.24	330	330
11	badffp	11.44	10.17	330	390
7	llama	16.47	14.30	450	450
6	libacop	16.66	15.31	**630**	607
4	libacop2	16.88	16.60	**630**	605
1	miplan	**20.22**	**21.88**	618	**779**