

Performance Analysis of Planning Portfolios

Sergio Núñez and **Daniel Borrajo** and **Carlos Linares López**

Departamento de Informática, Universidad Carlos III de Madrid
 Avda. de la Universidad, 30. 28911 Leganes (Madrid). Spain
 sergio.nunez@uc3m.es, dborrajo@ia.uc3m.es, clinares@inf.uc3m.es

Abstract

In recent years the concept of sequential portfolio has become an important topic to improve the performance of modern problem solvers, such as SAT engines or planners. The PbP planner and more recently Fast Downward Stone Soup are successful approaches in Automated Planning that follow this trend. However, neither a theoretical analysis nor formal definitions about sequential portfolios have been described. In this paper, we focus on studying how to evaluate the performance of planners defining a baseline for a set of problems. We present a general method based on Mixed-Integer Programming to define the baseline for a training data set. In addition to prior work, we also introduce a short empirical analysis of the utility of training problems to configure sequential portfolios.

Introduction

The AI community constantly designs faster and more efficient heuristics and algorithms for solving Automated Planning problems. However, it has not been able to develop a single planner that dominates all others for classical Automated Planning (Roberts and Howe 2009). Also if a planner does not solve a planning task quickly, it is likely that it will not solve it at all (Howe and Dahlgren 2002). Both facts led to a new concept of planner termed portfolio in the literature. This is based on the following idea: several planners are executed in sequence with shorter timeouts, expecting that at least one of them will find a solution in its allotted time. In case of solving planning tasks optimally, the portfolio halts as soon as one planner finds a solution; otherwise, all planners are invoked and the best solution is picked up. This technique has been shown to be a successful approach in classical Automated Planning.

Recently, different approaches to build planner portfolios have been introduced. Fast Downward Stone Soup (FDSS) (Helmert, Röger, and Karpas 2011) and PbP (Gerevini, Saetti, and Vallati 2009) are some interesting examples. The first one is a sequential portfolio of domain-independent planners. There are two main versions of FDSS, one for optimal planning and another one for satisficing planning. FDSS was configured using a set of training

problems from all the past International Planning Competitions (those celebrated between 1998 and 2008)¹ and a set of planning algorithms. The algorithm employs a hill-climbing search algorithm in the space of portfolios. The search starts from an initial portfolio which assigns zero seconds to every planning algorithm. At each iteration, it generates a set of possible successors (which are, in turn, new portfolios) to the current portfolio. Each successor is generated by increasing the allotted time of one planner by a small ratio of the total time. The best successor is selected as the current portfolio for the next iteration. Once the total time has been reached, the algorithm halts and the configuration of the portfolio is the best successor found so far.

The PbP planner is a portfolio-based planner with macro-actions, which automatically configures a sequential portfolio of domain-independent planners. The configuration relies on some knowledge about the performance of the planners in the portfolio for a specific domain and the observed utility of automatically generated sets of macro-actions. Nevertheless, PbP can be used without this additional knowledge. Thus, the portfolio schedules all planners by a round-robin strategy and it assigns the same time slot to the planners after being randomly sorted. When PbP uses domain-specific knowledge, it only selects a cluster of planners (which are sorted by performance) for configuring the portfolio. On the other hand, macro-actions sets are not always considered. Instead, they are taken into account only if they improve the performance of the planner for the domain at hand.

The most common approach to assess the performance of a new sequential portfolio consists of comparing its score with the score achieved by all entrants of the last IPC. We claim that this comparison is not enough as a metric to evaluate the performance of new portfolios because it does not show how far the portfolio is from the optimal configuration achievable for this particular set of planning tasks. Thus, we propose a general method based on Mixed-Integer Programming (MIP) to configure the optimal static sequential portfolio for some training data —i.e., problems and planners. The portfolio computed this way actually defines a baseline for the particular training data set considered. Using this baseline, the performance of any planner can be analyzed.

¹<http://ipc.icaps-conference.org>

To exemplify our technique, we have applied this approach to the sequential optimization (seq-opt) track of the Seventh International Planning Competition —IPC 2011.

Additionally, we have studied the utility of different problems to investigate whether it is necessary to configure a portfolio with data sets as large as possible or if, on the contrary, only a few should be considered. To carry out this analysis we have used the previous example, which configures the optimal static sequential portfolio for the planning tasks in the seq-opt track of the IPC 2011.

This document is organized as follows: first in Section 2, we define the concept of the static sequential portfolio. In Section 3 we show the motivations of our work and we describe our contributions. In Section 4 we describe the MIP model. Next, we show the experimental results of our approach over the planning tasks of the seq-opt track of the IPC 2011. Finally, in Section 6 we present our conclusions and introduce future work.

Portfolio Framework

In this work, we focus on static sequential portfolios for classical Automated Planning. A static sequential portfolio can be defined as a configuration of planners, where each one has been assigned a slice of time. Formally, a static sequential portfolio is a sorted set C of pairs $c_i = \langle p, t \rangle$, where:

- p is a planner, defined as a tuple (A, H, P) where A is a non-empty set of search algorithms (A^* , Hill-Climbing, ...), H is a non-empty set of heuristics (Merge&Shrink (Helmert, Haslum, and Hoffmann 2007), Delete-Relaxation (Hoffmann 2003), ...) and P is the policy which chooses which search algorithm $a_i \in A$ and heuristic $h_i \in H$ shall be used.
- Finally, t is the time allotted to planner p .

This sort of portfolios are termed as *sequential* since it is assumed that all planners are executed sequentially, i.e., it is never allowed to invoke more than one planner at the same time.

Motivation

Currently, the field of sequential portfolios is a hot topic. There are many open issues, like the anytime nature for satisfying planning, the order of the execution sequence, etc. We focus on studying how to evaluate the performance of planners and subsequently on analyzing the utility² of training problems to configure sequential portfolios.

Usually, the performance of new planners (and among them, sequential portfolios) is assessed by facing them up with the planning tasks of the last IPC. But this comparison does not show how good a portfolio is. It only shows whether the portfolio behaves either better or worse than entrants of the last IPC in that particular set of planning tasks. Even if it is better than the winner, this measure does not

provide any information on the relative quality of the portfolio. We propose instead to define the best possible sequential portfolio for a set of problems with an approach based on Mixed-Integer Programming so that all planners (including portfolios) can be compared against the same baseline.

Basically, to evaluate a solver (which can be either a new single planner or a sequential portfolio of new or existing planners), we propose to define the best possible sequential portfolio for a particular set of problems and to invoke it over the same set of problems. If a solver reaches a score higher than the best possible portfolio, the solver can be said to be an improvement, otherwise the quality of the solver can be proven to behave either worse or equally than just a linear combination of existing planners. Thus, it would not be judged as a real improvement over the current state of the art.

With the aim of configuring sequential portfolios most approaches are based on two sets of training data: one for planners, and another for training problems. Usually, the first one is a small set of heterogeneous and modern planners (e.g., PbP incorporates seven planners and the optimal version of FDSS considers eleven planners), though in some approaches new solvers are built to configure the portfolio; the second one is often a large set of training problems from past IPCs (as a matter of fact, FDSS used a total of 1163 training instances from the IPCs ranging from 1998 to 2008). Clearly, the selection of these instances is very relevant to the resulting quality of the portfolio. However, there is neither an algorithm to find the best training problem set nor a definition of what the best training problem set is. In this paper, we address these issues.

Furthermore, we would like to analyze the following question: is it necessary to process large sets of planning tasks to configure a portfolio? In other words, do all problems provide the same information or utility? According to our intuition, a good training problem set should contain only a few training instances with high utility. In this regard, we propose a classification of problems into three categories sorted by difficulty. The first one contains problems of high difficulty. These are expected to have no utility because no planner is able to solve them. The second category, labeled as medium difficulty, is composed of problems that are solved only by a few planners. As will be shown later, these problems are the most useful ones. The third category contains problems with limited utility, because most planners solve them all.

Mixed-Integer Programming model

MIP is the best choice for our purpose for three reasons. First, the search of the optimal static sequential portfolio from a set of training data (planners and problems) can be modeled as a MIP task. Second, this technique ensures optimal solutions. Third, MIP is fast, efficient and it eases the creation and modification of constraints, which brings flexibility. In the end, a linear combination of existing planners is a reasonable threshold for judging whether a new portfolio results in an improvement or not.

²We consider the utility of training problems to be the knowledge provided to configure high-performance portfolios as discussed next.

We have used GLPK³ for solving the MIP task introduced in this section. This package supports the GNU MathProg modeling language⁴, which divides our model into four sections: *Parameters*, *Variables*, *Objective function* and a number of *Constraints*.

The training data set contains a set S of planners ($|S| = n$) and a set I of training instances ($|I| = m$). The *parameters* store the input data that is generated by processing the output of the execution of every planner $p \in S$ with every instance problem $i \in I$. In particular, the following *parameters* have been defined in the MIP model:

- $q(p, i)$. Normalized plan quality found by the planner $p \in S$ for the training problem $i \in I$. If the planner p does not solve the problem i , the value of plan quality is set to zero.
- $r(p, i)$. Normalized time spent by the planner $p \in S$ to solve the training problem $i \in I$. If the problem i is not solved by the planner p in a given time, the parameter value will be set to the allotted time limit to solve the training problem i by the planner p .
- $m(p, i)$. Normalized maximum memory used by the planner $p \in S$ to solve the training problem $i \in I$.

As in every MIP problem, *decision variables* are defined to get the outcome of the problem. In this model, the following have been defined:

- $solved_by_{pi}$. While the same problem can be solved by an arbitrary number of planners only one is considered to be part of the portfolio discarding all the others as discussed below. To this end, this variable reports whether the plan found by the planner $p \in S$ for the training problem $i \in I$ is considered to configure the sequential portfolio or not. Therefore, it is a binary variable that takes values in the set $\{0, 1\}$.
- $quality_i$. Plan quality found by the sequential portfolio for the training problem $i \in I$.
- $time_p$. It is the output variable, which shows the allocated time to each planner $p \in S$ in the sequential portfolio.
- $memory$. Maximum memory used by the sequential portfolio.

With the purpose of defining the *objective function* of the MIP task, we have considered time, memory and plan quality for every training instance and every planner in the training data set. Time and memory are computed as the time spent and the memory used by a particular planner to solve a specific problem, divided by the time and memory bounds. Note that because values are normalized, the *objective function* is no referred to any particular bounds of time and memory. Likewise, we compute the plan quality in the range $[0, 1]$ for each training instance. If the portfolio does not solve an instance, the plan quality of this problem is set to zero. Otherwise, the plan quality of a solved instance is computed as the lowest (i.e., better) solution cost found by

any planner in the training data set, divided by the best solution found by the portfolio. Hence, the objective function is defined as:

$$\begin{aligned} \text{maximize :} \quad & w_1 \left(\sum_{i=0}^m quality_i \right) \\ & + w_2 \left(1 - \sum_{p=0}^n time_p \right) \\ & + w_3 (1 - memory) \end{aligned}$$

While the function used for evaluating planners that took part in the last two IPCs only focuses on plan quality, we propose an *objective function* that makes it possible to balance maximizing the score achieved by the portfolio (i.e., the number of problems solved or, equivalently, the *coverage* in the case of the sequential optimization track), the time spent and/or the memory used with different weights w_1 , w_2 and w_3 . Thus, using the *objective function* shown above it is possible to maximize the quality of the obtained solutions, as well as the performance of the sequential portfolio measured in time and memory consumption. For instance, with $w_1 = 1$, $w_2 = 0$ and $w_3 = 0$, only the overall score is optimized. However, if we increase w_2 and/or w_3 , a sequential portfolio is obtained that achieves the optimal score while it improves the time spent and/or memory used by the portfolio.

We have defined a set of *constraints* in the MIP model. *Constraints* (1) and (2) set the time and memory bounds to solve each training instance:

$$\sum_{p=0}^n time_p \leq 1 \quad (1)$$

$$memory \leq 1 \quad (2)$$

The MIP task analyzes plan quality, time and memory used by all planners for all training problems. For each of these problems, the task should select at most one planner to solve it. To avoid selecting two or more planners to solve the same instance, *constraint* (3) is used. It limits the number of planners that are considered to solve each training problem:

$$\sum_{p=0}^n solved_by_{pi} \leq 1, \forall i \in I \quad (3)$$

The *constraints* discussed next compute the sum of all time bounds allotted to all planners in the portfolio (4), the total score achieved by the portfolio (5) and the maximum memory used to solve a training problem (6):

$$time_p \geq solved_by_{pi} \times r(p, i), \forall i \in I, \forall p \in S \quad (4)$$

$$quality_i = \sum_{p=0}^n solved_by_{pi} \times q(p, i), \forall i \in I \quad (5)$$

$$memory \geq solved_by_{pi} \times m(p, i), \forall i \in I, \forall p \in S \quad (6)$$

Experimental setup and results

In order to define a baseline for the sequential optimization track of the last IPC, we have applied our MIP model over

³<http://www.gnu.org/software/glpk/>

⁴www.cs.unb.ca/~bremner/docs/glpk/gmpl.pdf

all problems I and planners S considered there. This track consists of 14 domains⁵ with 20 problems each. In total, there were 280 planning tasks.

When considering all the planners S that took part in the sequential optimization track of the IPC 2011, the portfolios have been removed from the selection and their solvers have been added instead. In particular, the two variants of Fast-Downward Stone Soup were discarded and the following solvers added as shown in Table 1: two versions of merge-and-shrink and the A* search algorithm with the blind heuristic.

Planner	Authors	Source
Blind	Silvia Richter, et al.	FDSS-2 planner
BJOLP	Erez Karpas, et al.	IPC 2011
CPT4	Vincent Vidal	IPC 2011
FD Autotune	Chris Fawcett, et al.	IPC 2011
Fork Init	Michael Katz, et al.	IPC 2011
Gamer	Peter Kissmann, et al.	IPC 2011
IFork Init	Michael Katz, et al.	IPC 2011
LM-cut	Malte Helmert, et al.	IPC 2011
LMFork	Michael Katz, et al.	IPC 2011
M&S-bisim 1	Raz Nissim, et al.	FDSS-1 planner
M&S-bisim 2	Raz Nissim, et al.	FDSS-1 planner
Selective Max	Erez Karpas, et al.	IPC 2011

Table 1: List of optimal planners used.

To generate the input data to our MIP model, we have executed every planner $p \in S$ with every instance problem $i \in I$ under the same conditions of the seq-opt track of the IPC 2011, this is, allowing each planner to solve every planning task with a time bound equal to 1800 seconds and a memory bound equal to 6 GB of memory. From all generated data, we have only considered the runtime $r(p, i)$, the maximum memory used $m(p, i)$, and whether the problem has been solved or not $q(p, i)$.

The score defined in the seq-opt track of the IPC 2011 only considers the number of solved problems. However, we also want to optimize the time spent by the sequential portfolio, while solving the higher number of problems. Since we cannot know in advance the best configuration of the weights w_1 , w_2 and w_3 , we have run a number of tests with different configurations and have picked up the best one.

The results are shown in Table 2. Good values for our target are $w_1 = 1$, $w_2 = 0.04$ and $w_3 = 0$. This configuration solves 200 instances and spends less time than other configurations that also solve 200 problems. Therefore, we have selected these weights to configure the optimal static sequential portfolio for the seq-opt of the IPC 2011, which is shown in Table 3.

Note that the MIP model used here does not specify any particular order to execute the planners. It only assigns an execution time to each planner, which is either zero or a positive amount of time. The definition of the execution se-

⁵These domains are: barman, elevators, floortile, nomystery, openstacks, parcprinter, parking, pegsol, scanalyzer, sokoban, tidybot, transport, visitall and woodworking.

quence is arbitrary and it is based just on the order in which the planners were initially specified. However, given that the optimal track of the IPC only considers coverage, the order does not affect the result.

The portfolio shown in Table 3 is the best static sequential portfolio w.r.t coverage. Furthermore, the time spent for solving the problems set has been optimized as well. However, since we have not tried all the feasible values of the weights w_1 , w_2 and w_3 , we do not know whether there is another linear combination of planners that solves the same number of problems spending less time. In order to check if there is such a combination, we have performed the following experiment:⁶

- Solve the MIP task setting $w_1 = 1$, $w_2 = 0$ and $w_3 = 0$.
- Let Q be the resulting value of the objective function for the found configuration.
- Add $\sum_{i=0}^m quality_i \geq Q - 0.001$ to the constraint set of the MIP model. It means the model can then only generate solutions that have a quality of at least Q (-0.001 is used for rounding errors).
- Solve the MIP task setting $w_1 = 0$, $w_2 = 1$ and $w_3 = 0$.
- Let T be the resulting value of the objective function for the found configuration.
- Add $1 - \sum_{p=0}^n time_p \geq T - 0.001$ to the constraint set of the MIP model.
- Solve the MIP task setting $w_1 = 0$, $w_2 = 0$ and $w_3 = 1$.

In the end, the portfolio that resulted from this method is exactly the same one generated by the original MIP model with $w_1 = 1$, $w_2 = 0.04$ and $w_3 = 0$.

FDSS solved 185 problems in the last IPC and this result is very likely to be used in the future to decide whether a new portfolio is better or not. However, as already claimed, it does not show how far the portfolio is from the optimal configuration for this particular set of planning tasks. Instead, we propose to use the number of problems solved by a linear combination of the entrants of the IPC 2011 as the score to beat in practice. This results from the consideration that all planners in the last IPC are state-of-the-art planners and that a linear combination of their performance is a reasonable estimator of the expected performance of state-of-the-art planners. In other words, solving less than 200 problems is not judged as a significant improvement since this number of problems can be solved in practice with a linear combination of existing planners.

Empirical analysis of the utility of training problems to configure sequential portfolios

Once the optimal static sequential portfolio for the seq-opt track of the IPC 2011 has been configured, we turn now our attention to the utility of training problems. To perform this task we have defined twelve sets of training problems, where each one contains instances from the seq-opt track of the IPC 2011 that were solved by a maximum number of planners.

⁶We thank a reviewer of a previous version of the paper for suggesting it.

w_1	w_2	w_3	Solved problems	Time	Memory
1.00	0.00	0.00	200	1800	5696.67
1.00	0.04	0.00	200	1705	5696.67
1.00	0.04	0.20	171	829	619.76
1.00	0.04	0.40	169	825	482.60
1.00	0.04	0.60	169	825	482.60
1.00	0.04	0.80	161	339	319.29
1.00	0.04	1.00	161	339	319.29
1.00	0.08	0.00	199	1563	5696.67
1.00	0.08	0.20	179	537	826.90
1.00	0.08	0.40	165	360	482.60
1.00	0.08	0.60	165	360	482.60
1.00	0.08	0.80	161	339	319.29
1.00	0.20	0.00	186	548	5380.70
1.00	0.20	0.20	166	265	778.61
1.00	0.20	0.40	163	345	385.74
1.00	0.20	0.60	161	338	319.29
1.00	0.20	0.80	161	338	319.29
0.80	0.00	0.00	200	1800	5696.67
0.80	0.04	0.00	199	1563	5696.67
0.80	0.04	0.20	171	829	619.76
0.80	0.04	0.40	169	825	482.60
0.80	0.04	0.60	161	339	319.29
0.80	0.04	0.80	161	339	319.29
0.80	0.04	1.00	161	339	319.29
0.80	0.08	0.00	186	548	5318.52
0.80	0.08	0.20	167	364	619.76
0.80	0.08	0.40	163	349	381.86
0.80	0.08	0.60	161	339	319.29
0.80	0.08	0.80	161	339	319.29
0.80	0.20	0.00	183	463	5380.70
0.80	0.20	0.20	165	234	778.61
0.80	0.20	0.40	157	190	381.86
0.80	0.20	0.60	155	179	319.29
0.80	0.20	0.80	152	180	247.56
0.60	0.00	0.00	200	1800	5696.67
0.60	0.04	0.00	199	1563	5696.67
0.60	0.04	0.20	165	360	482.60
0.60	0.04	0.40	161	339	319.29
0.60	0.04	0.60	161	339	319.29
0.60	0.04	0.80	161	339	319.29
0.60	0.04	1.00	161	339	319.29
0.60	0.08	0.00	186	548	5318.52
0.60	0.08	0.20	165	360	482.60
0.60	0.08	0.40	161	339	319.29
0.60	0.08	0.60	161	339	319.29
0.60	0.08	0.80	161	339	319.29
0.60	0.20	0.00	177	334	5305.70
0.60	0.20	0.20	157	190	381.86
0.60	0.20	0.40	155	179	319.29
0.60	0.20	0.60	152	180	247.56
0.60	0.20	0.80	152	180	247.56

Table 2: Results of our model for different configurations. For each configuration, the table shows the number of solved problems, the time spent (seconds) and memory used (MB) by the resulting sequential portfolio.

Planner	Allotted time (s)
CPT4	1
LM-cut	55
M&S-bisim 1	138
M&S-bisim 2	170
IFork Init	104
Gamer	1237
Total Time	1705

Table 3: Configuration of optimal sequential portfolio.

The first set is composed of all problems that were solved by at most one planner. The second set consists of all planning tasks that were solved by at most two planners, and so on. We have defined twelve sets because there were precisely twelve solvers after removing portfolios and adding their solvers instead as explained above. Figure 1 shows the problem distribution for the twelve sets of training problems.

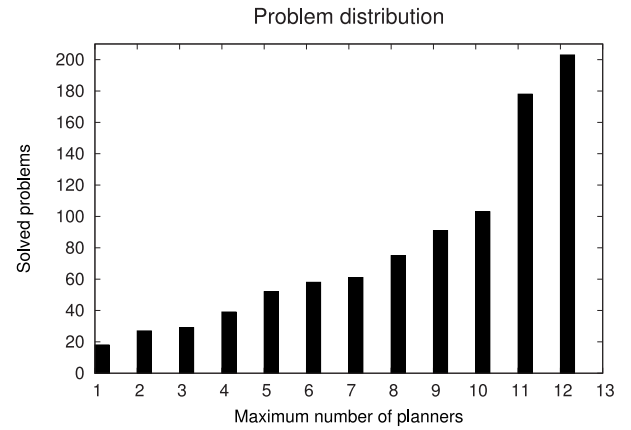


Figure 1: Problems distribution for the twelve sets of training instances.

We have executed our MIP model with each of the twelve training problem sets. The results are shown in Table 4. From the table, it results that the same performance (measured in coverage, time and memory) is achieved for all the sequential portfolios obtained using all sets of training problems but the first one (which consists just of those problems that were solved by at most one planner). The resulting sequential portfolio is shown in Table 5. Therefore, the minimum set of training problems necessary to configure the optimal portfolio for the competition contains only 27 problems (those which were solved by at most two planners). According to our previous classification, the first category consists of those problems solved by at most one planner; the second category consists of those solved by at most two planners and the third category is made of those planning tasks that were solved by three or more planners. This fact empirically confirms our initial intuition: not all training problems have the same utility and the best set of training problems should contain only a few instances with a high

utility.

Max. Planners	Solved problems	Time	Memory
1	16/18	1499	6144.00
2	24/27	1705	5696.67
3	26/29	1705	5696.67
4	36/39	1705	5696.67
5	49/52	1705	5696.67
6	55/58	1705	5696.67
7	58/61	1705	5696.67
8	72/75	1705	5696.67
9	88/91	1705	5696.67
10	100/103	1705	5696.67
11	175/178	1705	5696.67
12	200/203	1705	5696.67

Table 4: Results of our model using the twelve sets of training problems. For each set, identified by the maximum number of planners that solve its instances, the table shows the ratio between number of solved problems and size of the problem set, the time spent (seconds) and memory used (MB) by the obtained sequential portfolio.

Planner	Allotted time (s)	
	First training set	Remaining sets
CPT4	1	1
LM-cut	0	55
M&S-bisim 1	0	138
M&S-bisim 2	157	170
IFork Init	104	104
Gamer	1237	1237

Table 5: Sequential portfolios obtained using the twelve sets of training problems.

Furthermore, we have performed an additional experiment to double check this result. It consists of repeating the experiments of Fast-Downward Stone Soup for the sequential optimization track applying the MIP task over all planning tasks defined in the IPC 2008. This way, we can compare the performance of FDSS with the portfolio obtained (which behaves as a baseline portfolio planner) and analyze the difference of solved problems between them.

We have executed the MIP model with all the planners considered in the design of FDSS. Table 6 lists all of them. On the other hand, instead of using all the 1163 instances from the IPCs in the range covering the period from 1998 to 2008 (as it was done to configure FDSS) we have considered only the 240 planning tasks chosen at the IPC 2008⁷. Besides, we have considered two different configurations of weights w_1 , w_2 and w_3 .

The first configuration of weights ($w_1 = 1, w_2 = w_3 = 0$) focuses only on maximizing the number of solved problems, while the second one ($w_1 = 1, w_2 = 0.04, w_3 = 0$)

⁷In this edition of the International Planning Competition there were 8 domains with 30 planning tasks each resulting in a total number of 240 planning tasks.

Planner	Authors
Blind	Silvia Richter, et al.
BJOLP	Erez Karpas, et al.
h^1 landmarks	Erez Karpas, et al.
h^{max} landmarks	Malte Helmert, et al.
LM-cut	Malte Helmert, et al.
M&S-bisim 1	Raz Nissim, et al.
M&S-bisim 2	Raz Nissim, et al.
M&S-LFPA 10000	Malte Helmert, et al.
M&S-LFPA 50000	Malte Helmert, et al.
M&S-LFPA 100000	Malte Helmert, et al.
RHW landmarks	Erez Karpas, et al.

Table 6: Training planner set used by Fast Downward Stone Soup.

optimizes the time spent by the sequential portfolio while preserving the same coverage. Since the resulting portfolio might use less than the available time bound of the IPC (1800 seconds) we show the configuration of two different portfolios (denoted as Portfolio1 and Portfolio2) for each combination of weights: the solution found by the MIP task (*original* solution) and the same portfolio where the remaining time is uniformly distributed among all planners chosen by the MIP task —*modified* solution. Tables 7 and 8 show both configurations.

Planner	Allotted time (s)	
	Original solution	Modified solution
Blind	11	31
BJOLP	5	25
H1	5	25
Hmax	29	49
LM-cut	1046	1066
M&S-bisim 1	126	146
M&S-bisim 2	434	454
Total time	1656	1796

Table 7: Sequential portfolio obtained using $w_1 = 1, w_2 = 0$ and $w_3 = 0$.

Planner	Allotted time (s)	
	Original solution	Modified solution
LM-cut	1046	1110
M&S-bisim 1	126	190
M&S-bisim 2	434	498
Total time	1606	1798

Table 8: Sequential portfolio obtained using $w_1 = 1, w_2 = 0.04$ and $w_3 = 0$.

Finally, we have experimented with the resulting portfolio of these configurations with all the planning tasks of the sequential optimization track of the IPC 2011. The results are shown in Table 9. Both portfolios solve the same number of problems as FDSS-1, as shown in Table 10. Again,

these results endorse the idea that it is not necessary to use a large number of problems to train a portfolio. Instead, a smaller number of more informative problems can be used. Presumably, the time necessary to solve the MIP task (which is only a few seconds for the configurations tried) is significantly smaller than the time necessary to traverse the state-space of portfolio configurations with a hill-climbing search algorithm as in the case of FDSS. Remarkably, the planners selected for Portfolio2 are the same than those picked in the configuration of FDSS-1 without bjolp. This is not surprising, however, since bjolp was ranked eight in the last IPC.

Portfolio	Solved problems	
	Original solution	Modified solution
Portfolio1	183	185
Portfolio2	183	185

Table 9: Score of Portfolio1 and Portfolio2 for IPC 2011 seq-opt track.

Portfolio	Solved problems
FDSS-1	185
FDSS-2	182

Table 10: Score of FDSS-1 and FDSS-2 for IPC 2011 seq-opt track.

Conclusions and future work

We have presented a general method based on Mixed-Integer Programming to define the baseline for a specific set of problems, against which the real performance of planners can be measured. We have applied this method over all problems of the sequential optimization track of the IPC 2011 and we have shown that the real challenge consists of generating portfolios/planners that are capable of solving more than 200 problems when being trained with this data set. The reason is that any portfolio solving less problems falls below the results achievable by a linear combination of state-of-the-art planners. In the future, we encourage researchers to show real progress by configuring their portfolios with the planning tasks available in the last IPC and to compare the performance on the same set of problems with the baseline derived here.

In addition to this prior work, we have performed an empirical analysis to study whether all training problems have the same utility to configure sequential portfolios or not. The results show that, according to intuition, not all problems have the same utility. We have derived exactly the same configuration when running the MIP task with either 27 problems or 280 problems. The key observation is that the problems to use are only solved by a reduced number of planners—two in our experiments. Besides, when conducting the same experiment with the planning tasks of the IPC 2008 it turned out that the same number of problems solved by the winner of the IPC 2011 was achieved presumably far faster. As a side effect, the technique discussed herein is an

alternative to one of the most popular approaches to automatically configuring portfolios.

In the future we will perform a theoretical analysis based on sensitivity analysis of training problems to determine the accuracy of this assumption.

Additionally, we will try to analyze the utility of sequential portfolios for different time bounds. Commonly, the sequential portfolios have been tested using always the same bounds in effect in the International Planning Competition (1800 seconds and 6 Gb or memory to solve each problem). Some sequential portfolios have shown high performance using this time limit. But if these sequential portfolios are executed with a different time bound, their performance is just unknown. In particular, we are interested in finding out whether it is possible or not to derive sequential portfolios that solve at least as many instances as all entrants in the same amount of time or even less.

Finally, we want to study the order of the execution sequence for the optimal static sequential portfolio. We have obtained a randomly sorted sequential portfolio that solves 200 problems of the IPC 2011 seq-opt track. To solve each of these problems, the portfolio spends at most 1705 seconds. This runtime depends on the execution sequence. Therefore, we will analyze whether there is an order of the execution sequence for spending, on average, the minimum execution time.

Acknowledgments

This work has been supported by the INNPACTO program from the Spanish government associated to the MICINN project IPT-370000-2010-8 and different Spanish research grants through projects TIN2008-06701-C03-03, TIN2011-27652-C03-02, TIN2010-08861-E and TRA2009_0080.

References

- 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 2009)*. AAAI.
- Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*, 176–183.
- Helmert, M.; Röger, G.; and Karpas, E. 2011. Fast downward stone soup: A baseline for building planner portfolios. In *ICAPS 2011 Workshop on Planning and Learning* 28–35.
- Hoffmann, J. 2003. The metric-ff planning system: Translating “ignoring delete lists” to numeric state variables. *Journal of Artificial Intelligence Research* 20:291–341.
- Howe, A. E., and Dahlman, E. 2002. A critical assessment of benchmark comparison in planning. *J. Artif. Intell. Res. (JAIR)* 17:1–3.
- Roberts, M., and Howe, A. E. 2009. Learning from planner performance. *Artif. Intell.* 173(5-6):536–561.