A Guide to Portfolio-based Planning

Mauro Vallati

Dipartimento d'Ingegneria dell'Informazione
Università degli studi di Brescia, Italy
mauro.vallati@ing.unibs.it

Abstract. In the recent years the field of automated planing has significantly advanced and several powerful domain-independent planners have been developed. However, none of these systems clearly outperforms all the others in every known benchmark domain. This observation motivated the idea of configuring and exploiting a portfolio of planners to achieve better performances than any individual planner: some recent planning systems based on this idea achieved significantly good results in experimental analysis and International Planning Competitions. Such results let suppose that future challenges of Automated Planning community will converge on designing different approaches for combining existing planning algorithms.
This paper reviews existing techniques and provides an exhaustive guide to portfolio-based planning. In addition, the paper outlines open issues of existing approaches and highlights possible future evolution of these techniques.

1 Introduction

Automated Planning is one of the most prominent AI challenges; it has been studied extensively for several decades and lead to many real-world applications (see, e.g., [7]). During the last decade, Automated Planning has achieved significant advancements. However, while several powerful domain-independent planners have been developed, none of them clearly outperforms all others in every known benchmark domain. These observations motivate the idea of configuring and exploiting a portfolio of planners to achieve better overall performance than any individual planner. Moreover, portfolios based approaches have been successfully applied to a number of combinatorial search domains, most notably the satisfiability problem [24].

Very recently, a number of planners based on portfolio approach have been developed, and achieved impressive results in the last editions of the International Planning Competition (IPC6-7) [3, 2]: they won, or got very close to, in every track they took part. These include the deterministic track, learning track and multicore track. Achieved results let us presume that the future of AI planning will not be only focused on developing new planning algorithms, like in the last decade, but specially on designing promising techniques for combining and exploiting existing planning systems.

This paper reviews existing techniques for configuring a portfolio of planning algorithms, in order to: (i) give an overview of the state-of-the-art of portfolio-based planners, (ii) describe the decisions that have to be taken during the configuration process and, (iii) stimulate the development of new high-performance planning frameworks based on this promising approach.

The remainder of the paper is organized as follows. Section 2 briefly introduces Automated Planning, algorithm portfolios and existing portfolio-based planners; Section 3 describes the steps of portfolio configuration; Section 4 gives the conclusions.

2 Background

This section introduces first, the definition of Automated Planning tasks; then, it describes the idea behind portfolio-based approaches and finally, it presents the existing planning systems based on portfolio approach.

2.1 Automated Planning

Automated Planning studies the selection of actions in a dynamic system to reach a state of the system that satisfies a number of goals. Most of the approaches to Automated Planning assume that the system is deterministic, static, finite, and fully observable [7]. It is commonly described as ($\Sigma = (S, A, \gamma)$), where S is a finite set of states, A is a finite set of actions and $\gamma(s, a)$ is a single state when a is applicable to s.

According to this model, a *planning problem* can be defined as a tuple $\mathcal{P} = (\Sigma, s_0, g)$ where s_0 is an initial state and g corresponds to a set of goal states.

Solving a planning problem $\mathcal{P}$ consist of generating a plan, a sequence of actions $(a_1, a_2, \cdots, a_n)$ corresponding to a sequence of state transitions $(s_0, s_1, \cdots, s_n)$ such that: action a_i is applicable in state s_{i-1}, the state s_i is the result from executing a_i in s_{i-1} and s_n is a state where all goals are satisfied, $s_n \in G$.

The described model is called *classical planning*. Some assumptions made can be relaxed to study more expressive planning tasks. For example, *temporal planning* studies planning problems with durative actions. Typically, the objective of these tasks is minimizing the makespan of the plan, i.e., the difference between the start and end of the plan. *Planning under uncertainty* studies how to tackle planning problems when states are not fully observable and with non deterministic actions effects. *Planning with continuous actions* studies the planning task when the set of states is not finite because the effects of actions are continuous. *Planning with extended goals* studies how to generate plans when goals express requirements of different strengths, like users preferences. Each of these planning models has its own language extensions for representing the corresponding dynamic system, initial state, goals and solutions. Likewise, each model has its own algorithms for effectively solving the corresponding planning problems. On the whole, one can say that state-of-the-art planners often rely on heuristic search to generate the solutions [1].

2.2 Algorithm portfolios

The term *algorithm portfolio* was firstly introduced by Huberman et al. [13] to describe the strategy of running several algorithms in parallel. The idea was taken from economics, where portfolios are used to maximize a utility that has an associated risk. The algorithm portfolio approach was also studied by [8]. Several authors have since used

the term for describing any strategy that combines multiple algorithms, considered as black-boxes, to solve a single problem instance.

The space of algorithm portfolios include from approaches that use all available algorithms to approaches that always select only a single algorithm. The advantage of using the term portfolio to refer to this broader class of algorithms is that they all work for the same reason: select several algorithms in order to obtain improved performance in the average case.

2.3 Existing portfolio-based planners

In the field of Automated Planning, the idea of configuring and using a portfolio of techniques has been investigated by several researchers and has become a very interesting topic in the last few years. The first work on planner portfolios was done by Roberts and Howe [12, 20]; in this approach they generated a domain-independent portfolio of planners and compared different strategies for its configuration. It was not exactly a full automatic planning framework, but an in-depth study of the configuration and use of portfolios for (i) maximizing solved problems or (ii) minimizing runtimes.

Inspired by Roberts and Howe's work, but with several significant differences, Gerevini and collaborators developed PbP [5] (and lately, an enhanced version called PbP2 [6]); this planner extracts additional knowledge about the given domain and automatically configures a domain-specific portfolio of planners. Both versions of PbP are able to configure two different portfolios: one focusing on speed and the other focusing on plan quality, in terms of number of actions.

Fast Downward Stone Soup (here abbreviated FDSS) [10] is a recent approach to selecting and combining a set of forward-state planning techniques. Very recently, an extended version of FDSS (from now on, FDSS2) has been proposed [21]. The planner portfolio of this system consists of several different automatic-obtained configuration of a single high-performance planner, Fast Downward [9].

ArvandHerd [22] is a very recent pure parallel portfolio that simultaneously runs on different cores an instance of the well known domain-independent planner LAMA [19] and a set of instances of the random walk domain-independent planner Arvand [14]. In the multicore track of the last IPC [2] there were several planners based on the idea of running simultaneously different planning algorithms. For the purposes of this paper they are all similar, and we selected only ArvandHerd, the winner of the track, for representing the category.

Finally, a portfolio approach [17] has been used by the organizers of the IPC-7 ([2]) for evaluating the state-of-the-art of domain-independent planners. They presented a general method based on linear programming to define the baseline sequential portfolio for a specific set of problems, against which the real performance of planners can be measured and evaluated.

3 Portfolio configuration

In this section we will analyze every step of the portfolio configuration process for planning, with particular reference to existing systems (described in section 2.3). We will

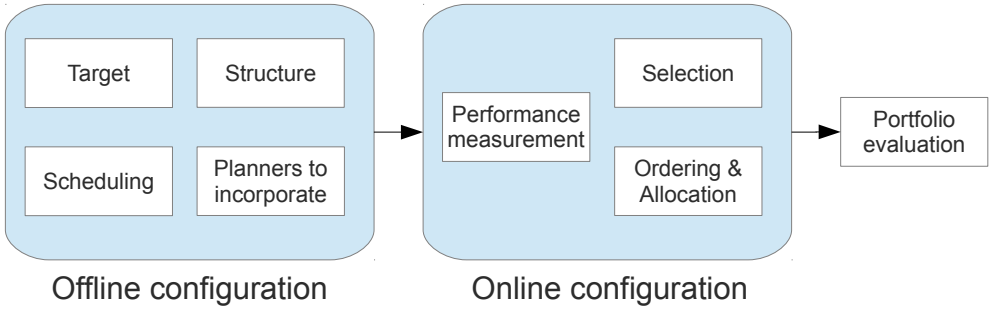

Fig. 1. An overview of steps required for configuring a portfolio of planners. Terms *Online* and *Offline* are considered w.r.t. learning instances.

consider the typical machine learning approach for extracting additional knowledge: the portfolio will be configured on a set of learning problems, easier than testing ones. Configured portfolio will be then used on much harder testing instances.

Fig. 1 gives a high level description of steps required for configuring a portfolio of planners. We divided the steps in two main sets: decisions to take *offline* and decisions to take *online*, w.r.t. the performance achieved by incorporated planners on learning problems used for the portfolio configuration. The former group is composed by the definition of the objective of the portfolio, the overall structure, the planners to consider and the scheduling strategy for running selected planners; the latter includes the performance measurement of planners on learning problems, the selection of promising planners, their ordering and CPU-time allocation to selected planners. Finally, we included also the evaluation of performances of configured portfolio on a subset of testing problems.

It should be clear that most of the phases are strictly related, and they do not have a clear predefined ordering. In this analysis we will describe each step individually and sequentially, in order to give the clearest representation of the whole configuration process.

3.1 Target and scope

A portfolio of planners is configured for optimizing a predefined objective function. Typically these functions are very easy and concern three different performances, usually taken individually: runtimes, quality of solution plans (in terms of number of actions or actions cost) and number of solved problems. A classical target, that is often required in IPCs is to maximize the solutions quality.

From the scope point of view, we can identify three different categories of portfolios: (i) domain-independent, (ii) domain-specific, and (iii) instance-specific.

A portfolio in category (i) is aimed at obtaining good mean performances on every possible benchmark domain; it is very general and, obviously, it can not exploit domain-specific knowledge. This is the case of FDSS, FDSS2 and ArvandHerd. On the contrary a domain-specific portfolio (PbP, PbP2) is configured for solving only problems from the given domain, it should have great performances on the specific domain and it can exploit additional knowledge (e.g., macro-actions [16]). Orthogonally to the previous approach, an instance-specific portfolio is created for solving problems that are "similar", either from different domains. It is usual, for evaluating the similarity of planning problems, to extract the value of some features related to the specific instance (e.g. number of objects), to the domain (e.g. number of operators) or to the performance of some planners (e.g., length of a relaxed plan). Instance-specific portfolios could theoretically be either domain-independent or domain-specific, but it is usual to think about them as a more sophisticated version of domain-independent approaches. Domain-specific approaches usually have very good performances on the selected domain, and it does not worth to look for further improvements.

3.2 Structure

Using a terminology close to the one introduced by Xu et al. in [24], we define an (a, b)-of-n portfolio as a set of n incorporated planners and a technique for selecting among them at least a and no more than b algorithms to be executed. For brevity, we also use the terms a-of-n portfolio to refer to an (a, a)-of-n portfolio, and n-portfolio for an n-of-n portfolio. This terminology includes a wide range of portfolios, for instance it includes also 1-of-n portfolios that are commonly defined as *algorithm selection* [18] frameworks, but it is very helpful for clearly describing existing approaches.

Recalling existing portfolio-based planners: PbP (PbP2) has a (1,3)-of-n structure, ArvandHerd has n-of-n structures. FDSS is a bit more complex to categorize since the number of selected planners is defined by an heuristic algorithm; the most correct way for describing its structure is (1, n)-of-n. FDSS2 exploits and compares different structures; like in FDSS it relies on heuristics algorithms for some of them, but it is also able to use, like ArvandHerd, all the included planners together; it goes from a (1, n)-of-n to n-of-n, depending on the selected approach for combining planners.

3.3 Planner scheduling

Portfolios can be *parallel* (all algorithms are executed concurrently), *sequential* (the execution of one algorithm only begins when the execution of previous algorithm has ended), or *mixed* (some combination of parallel and sequential).

In parallel portfolios (like ArvandHerd), there are enough CPUs for running all the selected planners in pure parallel. The portfolio ends when a planner finds a solution, or all the planners have spent the maximum available time. While it seems in principle very easy to implement, it becomes complex to deal with planners that share informations. It is the case of multiple planners exploring different area of the search space in parallel.

On the contrary, sequential portfolios run all the selected planners on a single CPU (FDSS). This strategy executes the planners to their maximum allotted time and quits at the first success or after all planners have spent their time. While it is easy to implement, this strategy requires refined techniques for estimating the amount of CPU time to allot to each planner. Moreover, if the portfolio's target is minimizing runtime, it is crucial to find the best order among selected planners.

Finally, a mixed strategy tries to mix the two previous techniques. This is usually done by "simulating" parallelism on a single CPU; for instance this could be done by using Round-Robin scheduling like in PbP or in one of the approaches studied by Roberts and Howe.

Obviously, there is not a clear limit to the combinations that is possible to obtain. It is theoretically possible, for instance, to configure a set of several sequential portfolios and execute them in parallel on different CPUs.

3.4 Planners to incorporate

One of the most important decisions to take while building portfolio-based planning systems, is choosing the planning algorithms to consider for the configuration of the portfolio.

The AI planning community constantly designs faster and more efficient heuristics and algorithms for solving Automated Planning problems. There is a large collection of domain-independent planners that can be considered while configuring a portfolio. The first temptation is, obviously, to consider *all* the available planners, like Roberts and Howe do in their study [12, 20]. This requires a dramatically high amount of CPU-time for evaluating the planners on learning problems (step described in section 3.5), therefore it is suitable only for the configuration of domain-independent portfolios, for which the evaluation step is done once.

The selection of planners incorporated in FDSS2 (and, similarly, in FDSS) is based on a completely different idea. In these works a single planner is selected, Fast Downward, with several different configurations that show to have high performance on some set of problems. It is an interesting approach that allows to achieve significant results; the Fast Downward planner is highly parametrized and includes numerous algorithms and techniques for planning that, correctly configured, work well on several different search space structures.

In the approach proposed in PbP2, the authors somehow combine the previous techniques by incorporating (i) a selection of state-of-the-art domain-independent planners, and (ii) a domain-specific configuration of the well known planner LPG [4]. The former is obtained by including all the planners that won an edition of the International Planning Competition, the latter by including the ParLPG planner [23].

Summarizing, it is important to include a large selection of uncorrelated planning techniques: including a very small set of algorithms will probably lead to poor performances; but on the other hand, including a lot of planners will take a remarkable amount of CPU-time for evaluating them on learning problems. Ideally, it would be perfect to include every existing different planning strategy.

3.5 Evaluation of the planners incorporated

This is generally the computationally most expensive step in the configuration of a portfolio.

Firstly, one must select learning instances on which evaluating incorporated planners. For configuring domain-independent portfolios, it is common practice to use a set of IPCs benchmarks domains and problems; that is helpful because they have been generated by human experts and, moreover, there exist official results for a preliminary evaluation of their hardness. On the contrary, for configuring domain-specific portfolios, are typically used random generators with some parameters to tune the problems difficulty; by working in this way it is possible to finely set the hardness of problems.

In order to evaluate the incorporated planners on the selection of learning problems, the performance metrics must be defined. It is usual to measure whether a plan is found (success or failure), the runtime needed for finding solutions and the quality of solutions. All of them are useful for configuring a portfolio optimizing any target function, as described in section 3.1.

Because each planner has its own way of declaring success, it is important to develop code to automatically extract these metrics from the output. Moreover, if incremental planners[1] are incorporated, it will be essential to define the way for measuring their performances. In PbP, for instance, the authors handle this by measuring the quality of all the solutions generated for a problem, and the corresponding needed CPU times.

3.6 Planners selection

Selecting the planners to include in the portfolio is strictly related with the number of incorporated planners and the maximum allowed size of the configured portfolio. This step could be useless in some portfolio structures: n-of-n (ArvandHerd and one of the configuration strategies included in FDSS2) design does not require any selection. The configured portfolio includes all the incorporated planners, and is based on the hypothesis that typical planners either solve a problem quickly or not at all [11]. This strategy is reasonable when: (i) the number of incorporated planners is limited; (ii) all the incorporated planners have really good mean performances; (iii) the maximum amount of CPU time for solving a problem is quite large and, (iv) the target of the portfolio is not minimizing the runtime.

In most of the cases, it is necessary to select only a subset of all the incorporated planners. Since the number of possible portfolios exponentially increase with the allowed maximum size of the configured portfolio, it is often computationally impossible to offer an exhaustive comparison: in those cases the most convenient approach is using heuristics techniques. A large selection of heuristics have been exploited and compared in the FDSS and FDSS2 ([10, 21]) papers.

On the contrary, if the number of possible portfolios is limited, it is suggested to exhaustively compare all of them. The comparison can be done by a statistical analy-

[1] Planners that are able to output several different plans by finding an initial satisficing solution and improving then its quality.

sis, like in PbP and PbP2, or by evaluating the performances of planners using some metrics like, for instance, the IPC scores ([2]).

Finally, the selection done by Roberts and Howe in their work is based on a completely different idea. They select all the planners that solved at least a predefined percentage of learning problems. The configuration of the resulting portfolio is then obtained by ordering all the selected planners using different strategies.

3.7 Allocation strategies and planners ordering

In this step of the portfolio configuration, the CPU-time allocated to selected planners and planners execution order are computed w.r.t. the selected scheduling (as described in section 3.3). It must be noted that planners ordering is fundamental for portfolios focusing on speed, but irrelevant on portfolios with different target.

If parallel portfolios do not need complex techniques for allocating CPU time to selected planner, it is a critical step for portfolios with different scheduling (both serial and mixed): giving too much (low) CPU time to a planner, could significantly worsen performances.

Existing portfolios with mixed scheduling strategies, like PbP and PbP2, compute the CPU time to allocate for each included planner in the following way. For each integrated planner, PbP defines a sequence of increasing *planning time slots*, $\langle t_1, ..., t_n \rangle$. Each t_i is the CPU-time that will be allotted to the planner during the testing phase. A t_i is defined as the CPU time required to solve a training problem during the performance measurement phase in a percentage p_i of cases. The sequence of increasing percentages $\langle p_1, ..., p_n \rangle$ from which the planning time slots are derived is defined by the vector $\langle 25, 50, 75, 80, 85, 90, 95, 97, 99 \rangle$. The execution order of selected planners is defined by the increasing CPU-time slots associated with them, shortest first.

For serial portfolios, the classical strategy is to equally divide the maximum amount of CPU time through all the selected planners. This strategy, even though is very easy, has shown to achieve significant results in terms of quality of plans (FDSS2). FDSS and FDSS2 incorporate allocation strategies in several planners selection heuristic algorithms. In those cases the CPU time allocated to a planner is heuristically estimated during the portfolio composition.

In the work of Roberts and Howe [20] they do an experimental analysis for evaluating serial and mixed portfolios. In mixed strategy they use a techniques similar to PbP for allocating CPU time to selected planners, while planners ordering is done by predictive models like, for instance, predicted probability of success or predicted runtime. We should recall that Roberts and Howe's system is based on domain-independent instance-specific portfolio configuration, so they can extract a set of features from the new instance for an online configuration of the portfolio on testing problems. While evaluating serial portfolios, Roberts and Howe allocate to each selected planner its average CPU time to succeed or the predicted CPU time for solving the new instance.

3.8 Evaluating the portfolio

Typically, a portfolio is configured by evaluating the performances of incorporated planners on a set of learning instances, that are somehow related with the testing problems.

Since the portfolio has been configured on problems different from the one on which it will be used, it is essential to evaluate its performance on (a subset of) testing instances. A configured portfolio must achieve, at least, better performances than every individual incorporated planner, so it is good practice to compare against all of them. After that, the main questions are: (i) given the selected structure of the portfolio, did we correctly configure it? and, (ii) is the selected portfolio structure (defined by the offline decisions taken w.r.t. Fig. 1) suitable for our target and scope? For finding an answer to the former question, the best strategy is to compare the configured portfolio with an oracle: a portfolio with same structure but configured exactly on the testing problems[2]. For the latter question, it would be enough to compare against differently structured portfolios. It should be noted that selecting the most appropriate portfolio structure for this comparison is -at least- as difficult as selecting the preferred portfolio configuration. The most convenient strategy is to compare with state-of-the-art portfolio-based planners, e.g. by selecting them from recent IPCs.

4 Conclusions

The existing Automated Planning technology offers a large, growing set of powerful techniques and efficient domain-independent planners, but none of them outperforms all the others in every known planning domain. From a practical perspective, it is then useful to consider a portfolio-based approach to planning involving several techniques and planners: recently, several different high-performance portfolio-based planners have been developed.

Our review is motivated by the excellent results achieved by portfolio-based planning systems in recent International Planning Competitions: they won, or got very close to, in every tracks they took part. These impressive results let suppose that future Automated Planning challenges will be related to algorithms and techniques for effectively combining planners, in order to obtain results that can not be achieved by a single domain-independent planner.

In the paper we identified the different decisions that have to be taken for configuring a portfolio of planners, and divided them in two subsets: decisions to take *offline* and decisions to take *online*, w.r.t. the learning problems used for the portfolio configuration. In the former group there are choices to make before working on learning instances, while in the latter the decisions are related with the performance of incorporated planners on example problems. We exhaustively described every configuration step, and analyzed how existing approaches in planning deal with them.

4.1 Open Issues

Below, we provide with a list of what we consider to be open issues or future avenues in portfolio configuration for planning.

[2] This is exactly the strategy adopted by Núñez et al. [17] for generating a baseline of the performance, to compare with other planners.

Target As introduced in section 3.1, every portfolio must have a target function to optimize. Most of the existing approaches are optimized for finding good quality plans (in terms of number of actions or action costs) or for maximizing the number of solved problems, and all of them exploit configured portfolios composed by several different planners. The only existing system that is able to configure a domain-specific portfolio (also considering additional domain-related knowledge) for minimizing runtime is PbP (and its latest version, PbP2). Analyzing the runtimes-configured portfolios that PbP generated for IPC6-7 benchmark domains [3, 2], it is easy to note that usually a single planner (possibly with additional knowledge extracted from the domain under the form of macro-actions) is selected. It would be interesting, for all the Automated Planning community, to offer an in-depth analysis for better understanding this behaviour. Is it related with the scheduling strategy, with the other knowledge extracted from the domain or is it typical of domain-specific portfolios focusing on speed?

Planners selection A striking result showed in [21] is that, in terms of solution quality, none of the more sophisticated strategies for configuring portfolios, performs better than the uniform portfolio (i.e., *all* the incorporated planners are selected and have the same amount of CPU time). This result supports the assumption they made that most planners either solve a problem fast or not at all. Additionally, their work indicates that portfolio performance can be improved much more by diversifying the set of incorporated planners than by adjusting selected planners' runtimes. This result is very strong, and it seems that the portfolio configuration is critical only in the case we do not have enough different domain-independent planners or we have a very short CPU time for solving problems.

Learning problems Implementing mechanisms to autonomously collect learning examples for Automated Planning is still an open issue. Traditionally, training problems are selected from IPCs benchmark or obtained by random generators with some parameters to tune the problems difficulty. The former approach is limited to already existing domains and instances, that are few. The latter has two main limitations: (i) it is not trivial to guarantee problems' solvability; (ii) the generators' parameters are domain-specific and tuning these parameters to generate good quality learning examples implies domain expertise.

Predictive model The only existing work in Automated Planning that builds and exploits predictive models for configuring a portfolio is the one proposed by Roberts and Howe [20]; after that, the recent approaches have abandoned this way and obtained even so significant results. This seems to be counterintuitive with the results obtained by portfolio approaches in different fields of Artificial Intelligence (see, e.g. [24]), where predictive models are extensively and efficiently used.

Automated framework Most of the existing systems do not have a completely automated configuration process. It would be useful, for a better understanding of portfolios and planners performances, to have a framework that is able to automatically generate several different classes of portfolios and to compare all of them through different techniques. Such a framework will provide an easy tool for studying the performances of portfolios and to evaluate the impact of new ideas in configuration steps. Moreover, that framework would also suggest a potential method for test-

ing new planners, based on measuring the performance improvements obtained in several different portfolios by adding them as incorporated planners.
In different fields of AI already exist some tools like the one we just outlined. A full working example, that the Automated Planning community should regard, is the HAL system [15]. It has been designed for supporting the empirical analysis and design tasks encountered during the development, evaluation and application of high-performance algorithms.

Share informations Existing portfolio approaches use incorporated planners as black-boxes. Selected planners do not share informations, knowledge or evaluations about current problem. In order to push forward the performances of a portfolio of planners, we believe that they should share informations and cooperate for reaching the goal (e.g. by exploring different areas of the search space or by trying to satisfy independent goals).

This review focused on existing techniques for configuring a portfolio of planners in order to: (i) give an overview of the state-of-the-art of portfolio-based planners, (ii) describe the decisions that have to be taken during the configuration process and, (iii) stimulate development of new high-performance planning systems based on this approach.

Further studies are needed to analyze the highlighted open issues and to increase the performances that can be achieved by exploiting a portfolio approach in Automated Planning. We are confident that these techniques, only recently applied in Automated Planning, will lead to further significant improvements in the close future.

References

1. Bonet, B., Geffner, H.: Planning as heuristic search. Artificial Intelligence 129, 5–33 (2001)
2. Coles, A., Coles, A., Olaya, A.G., Jiménez, S., Lòpez, C.L., Sanner, S., Yoon, S.: A survey of the seventh international planning competition. AI Magazine 33, 83–88 (2012)
3. Fern, A., Khardon, R., Tadepalli, P.: The first learning track of the international planning competition. Machine Learning 84, 81 – 107 (2011)
4. Gerevini, A., Saetti, A., Serina, I.: Planning through stochastic local search and temporal action graphs. Journal of Artificial Intelligence Research (JAIR) 20, 239 – 290 (2003)
5. Gerevini, A., Saetti, A., Vallati, M.: An automatically configurable portfolio-based planner with macro-actions: PbP. In: Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS-09). pp. 350 – 353 (2009)
6. Gerevini, A., Saetti, A., Vallati, M.: PbP2: Automatic configuration of a portfolio-based multiplanner. In: Working notes of 21st International Conference on Automated Planning and Scheduling (ICAPS-11) 7th International Planning Competition (2011)
7. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory & Practice. Morgan Kaufmann Publishers (2004)
8. Gomes, C.P., Selman, B.: Algorithm portfolios. Artificial Intelligence 126(1-2), 43–62 (2001)
9. Helmert, M.: The Fast Downward planning system. Journal of Artificial Intelligence Research (JAIR) 26, 191 – 246 (2006)
10. Helmert, M., Röger, G., Karpas, E.: Fast Downward Stone Soup: A baseline for building planner portfolios. In: Proceedings of the ICAPS-11 Workshop of AI Planning and Learning (PAL) (2011)

11. Howe, A., Dahlman, E.: A critical assessment of benchmark comparison in planning. Journal of Artificial Intelligence Research (JAIR) 17, 1 – 33 (2002)
12. Howe, A., Dahlman, E., Hansen, C., vonMayrhauser, A., Scheetz, M.: Exploiting competitive planner performance. In: Proceedings of the 5th European Conference on Planning (ECP-99). pp. 62–72 (1999)
13. Huberman, B., Lukose, R., Hogg, T.: An economics approach to hard computational problems. Science 265, 51–54 (1997)
14. Nakhost, H., Müller, M.: Monte-carlo exploration for deterministic planning. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09). pp. 1766–1771 (2009)
15. Nell, C., Fawcett, C., Hoos, H.H., Leyton-Brown, K.: HAL: A framework for the automated analysis and design of high-performance algorithms. In: Proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION-5)
16. Newton, M.H., Levine, J., Fox, M., Long, D.: Learning macro-actions for arbitrary planners and domains. In: Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS-07)
17. Núnez, S., Borrajo, D., Lòpez, C.L.: How good is the performance of the best portfolio in ipc-2011? In: Proceedings of ICAPS-12 Workshop on International Planning Competition (2012)
18. Rice, J.R.: The algorithm selection problem. Advances in Computers 15, 65 – 118 (1976)
19. Richter, S., Westphal, M.: The LAMA planner: Guiding cost-based anytime planning with landmarks. Journal of Artificial Intelligence Research (JAIR) 39, 127 – 177 (2010)
20. Roberts, M., Howe, A.: Learned models of performance for many planners. In: Proceedings of the ICAPS-07 Workshop of AI Planning and Learning (PAL) (2007)
21. Seipp, J., Braun, M., Garimort, J., Helmert, M.: Learning portfolios of automatically tuned planners. In: Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12) (2012)
22. Valenzano, R., Nakhost, H., Müller, M., Schaeffer, J., Sturtevant, N.: Arvandherd: Parallel planning with a portfolio. In: Proceedings of the 20st European Conference on AI (ECAI-12) (2012)
23. Vallati, M., Fawcett, C., Gerevini, A., Hoos, H., Saetti, A.: Automatic generation of efficient domain-specific planners from generic parametrized planners. In: Proceedings of the 18th RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (2011)
24. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: SATzilla: Portfolio-based algorithm selection for SAT. Journal of Artificial Intelligence Research (JAIR) 32, 565–606 (2008)