Generating and Exploiting Cost Predictions in Heuristic State-Space Planning

Francesco Percassi, Alfonso E. Gerevini,[*] Enrico Scala, Ivan Serina
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Brescia, Italy

Mauro Vallati
Department of Informatics, University of Huddersfield
Huddersfield, UK

Abstract

This paper proposes and investigates a novel way of combining machine learning and heuristic search to improve domain-independent planning. On the learning side, we use learning to predict the plan cost of a good solution for a given instance. On the planning side, we propose a bound-sensitive heuristic function that exploits such a prediction in a state-space planner. Our function combines the input prediction (derived inductively) with some pieces of information gathered during search (derived deductively). As the prediction can sometimes be grossly inaccurate, the function also provides means to recognise when the provided information is actually misguiding the search. Our experimental analysis demonstrates the usefulness of the proposed approach in a standard heuristic best-first search schema.

Introduction

The performances of current planning systems are affected by the structure of the search space, which depends on the application domain and its encoding. In many cases, the planning performance can be improved by deriving and exploiting knowledge about the structure of the problem and its potential solutions that is not explicitly given in the input formalisation, and that can be used for optimising the planner performance. Well-known examples include portfolio configuration (Seipp et al. 2015), reformulation approaches such as macro-operators (Botea et al. 2005; Scala 2014; Scala and Torasso 2015), entanglements (Chrpa, Vallati, and McCluskey 2019), action schema splitting (Areces et al. 2014), and model configuration (Vallati and Serina 2018).[1]

Another kind of knowledge that could be derived by analysing a planning problem is the expected (predicted) cost of its solutions, and in particular of an optimal or good quality solution. Taking inspiration from the significant amount of work devoted at handling solution quality bounds (see, for instance, (Stern et al. 2014; Thayer and Ruml 2011)), the question that we address in this paper is the following: *How can we exploit a prediction of the cost of a (good) solution plan to improve search performance?* Predictions on the costs of solution plans can come from different sources: they can be made by human experts, or computed automatically through, e.g., inductive methods based on machine learning approaches. The use of such a prediction during the planning search poses a number of interesting challenges about how to fruitfully exploit it to improve planning performance. This is because predicted values can sometimes be grossly inaccurate and either under- or over-estimating the quality of the best plan, or of the best-quality plan that can be found by the considered planning approach within a certain time limit. Due to this, such predictions can not be straightforwardly used as bounds, but there is a need for appropriately designed approaches.

In order to address the above question, in this paper first we introduce a domain-independent approach for predicting the cost of a "good" solution of a planning instance; then, to exploit the knowledge provided by these predictions, we introduce a best-first search schema that chooses the nodes to visit considering the predicted solution cost combined with pieces of information collected during search (heuristic costs to reach the goal, current cost from the root, number of expanded nodes). In particular, we propose a heuristic function aimed at putting in synergy these different sources of information for improving performance. The heuristic is also provided with means to mitigate the impact of the prediction when the search recognises that such a prediction is actually misguiding the search because too inaccurate.

To evaluate the proposed techniques we perform an experimental analysis using well-known benchmarks from the international planning competitions, and we show that the addition of this knowledge can be beneficial in the context of a best-first search schema.

Predicting Plan Cost

Useful information about the structure of a planning instance can be extracted under the form of some "features". Each feature summarises a potentially important property of the considered instance. The whole set of features can be seen as the fingerprint of the planning instance at hand. Features have been exploited in planning mostly for predicting the

[*]Corresponding author. Email: alfonso.gerevini@unibs.it

[1]For an extensive overview of the field, the interested reader is referred to (Celorrio et al. 2012).

CPU-time needed by a given planner to solve a new instance (e.g., (Fink 1998; Howe et al. 1999; Roberts et al. 2008; Fawcett et al. 2014; Cenamor, de la Rosa, and Fernández 2016; Rizzini et al. 2017)). More recently, features have also been used as a measure to compare planning instances in terms of differences and informativeness (Cenamor and Pozanco 2019). This amount of research provided the community with a large number of features that can be automatically extracted from the specification of a planning instance.

Given the results achieved by Fawcett et al. (2014) in predicting planners' performance, we decided to consider the set of features they exploited, which are 311 different features for each planning instance. This set of features demonstrated to be useful for predicting the behaviour of a different range of planners. The features that Fawcett et al. exploited are obtained from the analysis of: (i) different encodings of a planning problem (PDDL, SAT, SAS+); (ii) pre-processing statistics of LPG (Gerevini, Saetti, and Serina 2011) and Fast Downward (Helmert 2006); (iii) the search space topology analysed by Torchlight (Hoffmann 2011), and (iv) Fast Downward probing features –brief runs of a planner on the considered instance, in order to extract information from its search trajectories. However, due to the significant runtime required to extract Fawcett et al.'s features from the SAT encoding of the planning instance, we decided to remove these features from our set. Therefore, our final set consists of 196 features that can be generally extracted in a matter of few CPU-time seconds.[2]

Our aim is to obtain a predictive model that is able to estimate the cost of a good solution of a given planning instance, on the basis of the extracted features. In order to develop such a model, here we trained it to predict the cost of the best known solution of a training planning instance. The data used to train the predictive model comes from a large set of planning domains. Indeed, we collected as many PDDL planning instances as possible. We included instances from the following sources: (i) IPC-98 and IPC-00; (ii) IPC deterministic tracks from 2002 to 2011; (iii) learning tracks of IPC-08 and IPC-11; (iv) FF benchmark library; (v) Fast Downward benchmark library; (vi) UCPOP Strict benchmarks, and; (vii) Sodor and Stek domains used by Roberts et al. (2008). In total, more than 7000 planning instances were used to derive the training data set.

As cost value to predict, for each planning instance we considered the best known quality (lowest cost) of a plan, whether available, by looking at the data stored in "Planning.Domains". Otherwise, we considered the best solution generated by a set of selected state-of-the-art planners using 1800 CPU-seconds for each run. Such planners are: LAMA (Richter and Westphal 2010), Fast Downward (Helmert 2006), LPG (Gerevini, Saetti, and Serina 2006), Madagascar (Rintanen 2012), FF (Hoffmann and Nebel 2001), Arvand (Nakhost et al. 2011), Probe (Lipovetzky and Geffner 2011). They were chosen because they exploit very different planning techniques, and can therefore provide solution plans of significantly different costs – maximising the possibility of obtaining good cost plans on the considered instances.

[2]The interested reader is referred to (Fawcett et al. 2014) for a detailed description of the considered features.

For generating the predictive model, we used the WEKA tool (Hall et al. 2009), and in particular Auto-WEKA (Kotthoff et al. 2017) for selecting the best technique to use and the best configuration of the corresponding parameters. Auto-WEKA was run for 2 CPU-time days in a 10-fold cross-validation on all the considered training instances. The resulting predictive model that we obtained uses additive regression models based on random forests.

Regarding the use of the model, the CPU-time needed to extract the considered features and the prediction made by the model take less than 0.1 CPU-time seconds.

Exploiting Potentially Inaccurate Predictions

A common approach to solving planning problems (Ghallab, Nau, and Traverso 2004) is using a forward state space planner, guided by a best first search such as A*. This section proposes a bound-sensitive heuristic function exploiting an externally provided estimation $B \in \mathbb{R}$ of the cost of a plan.

Similarly to standard wA*, we use a constant weight w for $h(n)$ and collect the cost $g(n)$ to the node n as the cost of the prefix that goes from the initial state/root node to n; moreover, we consider two heuristic estimates of the cost from n to the goal, namely $h_\delta(n)$ and $h_s(n)$. $h_\delta(n)$ is used for assessing whether the prefix under exploration has trespassed the estimate given as an input; $h_s(n)$ is used to guide the search. In particular, we intend $h_\delta(n)$ to be a robust estimator of the current situation, and so we usually prefer for an admissible estimate. Instead, for $h_s(n)$, we use an estimate that is not guaranteed to be admissible but that is potentially more informed. Our A* search process organises the exploration by expanding the node n in the frontier that minimises the cost function $f(n) = g(n) + w \cdot h^{Bound}(n)$ where

$$h^{Bound}(n) = h_{\mathrm{s}}(n) \cdot \left(\frac{B}{g(n) + h_\delta(n)}\right)^{(1-p_{rate})} \qquad (1)$$

and

$$p_{rate} = \frac{\#\{n \in \{expanded\ nodes\ \} \mid\ g(n) + h_\delta(n) > B\}}{\#\{expanded\ nodes\}}$$

Intuitively, the use of h^{Bound} accelerates the search towards the nodes that are close to B (the provided predicted cost), and from that point on-wards, it promotes an exploration that is more sensible to the g-values (the heuristic contribution is diminished in the minimisation). This is obtained by amplifying the contribution of the heuristic for all those nodes n with $g(n)+h_\delta(n) < B$, and diminishing the heuristic value of those nodes n with $g(n)+h_\delta(n) > B$. Note that, since $h_\delta(n)$ is an admissible function, $g(n) + h_\delta(n)$ plays the role of an optimistic predictor for the cost of the solution plan that the search would obtain through the candidate node n.

It is easy to see that the overall behaviour of the search heavily depends on how the bound B relates to the solutions space of the problem. When B is incorrectly too tight (much lower than the cost of the optimal cost), the second factor in (1) gets very small, and therefore we expect the search to

Approach	Cov.	avg-C	IPC-Q	PAR10	IPC-R
wA*	191	**248.4**	187.5	10443.0	166.7
wA*-bound	**228**	279.3	**215.2**	**8992.3**	**211.6**
GBFS+wA*	277	**729.2**	270.4	8265.1	230.3
GBFS+wA*-bound	277	766.0	**271.6**	**8084.1**	**264.1**

Table 1: Comparison of the performance of wA* using our bound-sensitive heuristic function (wA*-bound) and the baseline wA* alone (top table) or after a run of GBFS (bottom table). Results are presented in terms of coverage (number of solved instances), normalized average plan costs, IPC score quality, PAR10, and IPC score runtime.

expand more nodes; most of the search will be indeed done favoring nodes with lower g-values. Instead, when B is too large (much larger than the optimal cost), the second factor of (1) makes the search greedier, because it favours the indication provided by h_s. In general this can reduce the number of expanded nodes, but it can also compromise the quality of the obtained solution. Nevertheless, we experimentally observed that having B over estimating the optimal cost does not worsen solution quality significantly, especially when B is only slightly larger.

To overcome the issue related to the use of a too tight B (potential very high number of visited nodes), we devise a further exponential modifier in the definition of h^{Bound}: $(1 - p_{rate})$. The aim of this modifier is to alleviate the effect of the bound on the heuristic by taking advantage of the information acquired whilst searching. More precisely, p_{rate} is defined as the fraction of expanded nodes that have a value for $g(n) + h_\delta(n)$ (where h_δ is admissible) that exceeds bound B. In this way, the more expanded nodes exceeded B, the closer the second factor of (1) gets to one. As a matter of facts, when p_{rate} increases, the exponent of the second factor tends to zero and the second factor itself becomes irrelevant (i.e., close to 1).

It is important to remark that the described "adaptive" heuristic function h^{Bound} exploiting B can inform the search in a way that is in contrast with the heuristic evaluation of the original function h_s. For instance, suppose $B = 7$, $w = 1$ and there are two nodes on the search frontier $n1$ and $n2$ such that $h_s(n1) = 5$, $h_\delta(n1) = 3$, $g(n1) = 1$, $h_s(n2) = 9$, $h_\delta(n2) = 7$, and $g(n2) = 1$. wA* using h_s prefers $n1$ to $n2$, while wA* with h^{Bound} prefers $n2$ because $h^{Bound}(n1) = 5 \cdot \frac{7}{3+1} = 8.75$ and $h^{Bound}(n2) = 9 \cdot \frac{7}{7+1} = 7.875$. A situation like this could happen, for instance, in a logistic problem where we are at a location from which we can reach the target from two paths; one of such paths appears to be better according to h_s, but it is actually not viable to the end because along this path, differently from the other longer path, there is no refuel station, and the target location can be reached only if at least one refuel is done before a maximum number of moves. The predicted cost B could alter the h_s-values towards the longer but safer path.

Experimental Analysis

We implemented the bound-sensitive heuristic function in Fast Downward (Helmert 2006), taking into account the settings used by the well-known planner LAMA, that includes a wA* search episode with $w = 5$ alternately guided by a couple of inadmissible heuristics: h^{FF} (Hoffmann and Nebel 2001) and h^{LM} (Richter and Westphal 2010). Our h^{Bound} can use as its h_δ component any heuristic. We modified the planner to be guided by the proposed bound-sensitive heuristic, where $h^{\text{LM-Cut}}$ is used as h_δ, and either h^{FF} or h^{LM} as h_s according to LAMA's alternating policy.

In the experimental analysis, for each problem of the considered benchmarks (that are different from the problems used to train the predictor), the predictor is provided with features extracted from the problem, and it returns the predicted cost of solving such a problem. This predicted cost plays the role of B in Equation 1. The plan cost prediction made on a given planning instance can at times be grossly wrong and this can harm the search substantially. To partially overcome this problem, we employ a simple preprocessing that tests if the predicted value B is lower than the value of the admissible heuristic $h^{\text{LM-Cut}}$ (Helmert and Domshlak 2009) computed in the initial state; if B is lower, the prediction underestimates even the cost of a lower bound for an optimal solution for the problem considered, and so we do not use h^{Bound}. If instead B is higher or equal to the value of $h^{\text{LM-Cut}}$, then h^{Bound} is used.

We considered 26 domains from the satisficing track of IPCs 2014 and 2018, for a total of 520 instances; among them h^{Bound} was disabled (by the preprocessing) in 69 instances which are distributed over 6 of the 26 considered domains. These instances are removed from our evaluation.[3]

Our experiments were run on an Intel Xeon Gold 6140M CPUs with 2.30 GHz. For each instance we set a cutoff time of 1800 seconds, and memory was limited to 8 GB. Our evaluation considers *coverage* (number of solved problems), *average plan costs*, *PAR10* (Penalised Average Runtime), *IPC quality score*, and *IPC runtime score*, as defined in IPC-14.

Results

Table 1 summarises the results achieved by using h^{Bound} in a single run of wA* (top) and when run after a Greedy Best First Search (GBFS) episode (bottom), versus the baseline wA* and GBFS+wA* alone, respectively. It is easy to observe that, compared to the baseline wA*, wA*-bound can significantly boost the performance in terms of both coverage (+19%) and runtime. This comes at the cost of an average negative impact on the quality of the provided solutions (approx. 12.4%). Looking at the raw data that we collected for these two configurations, we observe a 22% reduction in nodes expansion over all benchmark problems, with results that vary domain by domain: in 9 out of the 26 considered domains, the use of the predicted cost decreases the number of expanded nodes (obviously considering the problems solved by both systems), while in 11 domains the number of expanded nodes is worsened. Coverage-wise, instead, wA*-bound solves more instances than wA* in 10 domains, while in 2 domains wA* solves more problems than wA*-bound. We can observe some complementarity between the system

[3]The actual predicted costs and the model parameterisation can be found at https://bitbucket.org/maurovallati/icaps-2020/

that uses the cost prediction, and the system that does not. Indeed, wA*-bound solves 47 instances that are not solved by the baseline, while wA* solves 10 instances that are not solved by wA*-bound.

The bottom part of Table 1 evaluates the performance of the proposed approach when used together with GBFS; The baseline here is GBFS+wA*, where our approach is GBFS+wA*-bound. Note that the second episode of search does not consider the solution found by the GBFS during the search: GBFS is used only to help increasing instance coverage performance, since it can be the case that GBFS can solve instances that wA* alone can not.[4] In both considered configurations, if the second episode of wA* (with or without bound) is not completed, we penalize the runtime by an amount equal to 1800 seconds. As shown in Table 1, GBFS+wA* tends to provide better quality solutions, but GBFS+wA*-bound reduces runtime.

To shed some light on the "value" of the cost predictions, we carefully analysed the domain-by-domain performance. The first aspect to acknowledge is that it is hard to define what should be the *perfect* prediction. One may argue that the cost of the optimal plan is a perfect prediction. However, our intuition is that the value of a prediction has to be related to the planning approach that has to exploit such prediction: it may be the case that using the optimal plan cost could make finding a solution significantly harder for a given search technique. For this reason, we decided to evaluate the quality of a prediction according to how close it is to the cost of a plan that would have been found by an episode of baseline wA* search ($w = 5$). Using this metric, we observed that for a large number of domains the predictions tend to be reasonably accurate: in 8 domains the prediction is proved wrong by less than 40%; in 5 domains, however, predictions overestimate by more than three times the cost of the solution found. Interestingly, it seems that a large overestimation does not always lead to worse performance. In GED and Maintenance, where predictions largely overestimate, the use of h^{Bound} in wA* is still beneficial. This is not the case in Spider, where the large overestimation leads instead to significantly worse performance. It may indeed be the case that the structure of the domain plays a pivotal role in the behaviour of h^{Bound} when B is a large over- or under- estimation. Out of the considered domains, in Barman, Childsnack, Hiking, Termes, Transport, and Visitall we observed that the predicted value tend to be an underestimation of the cost of the actual plan identified.

Figure 1 shows how the coverage of the considered techniques evolves over time. In particular, we considered the baseline wA*, wA*-bound, and the wA*-bound without p_{rate} in order to assess the importance of p_{rate} in the computation of h^{Bound}. On very small instances, solved in less than one CPU-time minute, there is no significant performance difference among the compared approaches; intuitively this is due to the fact that for instances quickly solved, the use of the predicted cost does not lead to the exploration of substantially different areas of the search space. When

[4]The cost of the GBFS solution (if any) is not used instead of B in h^{Bound}.

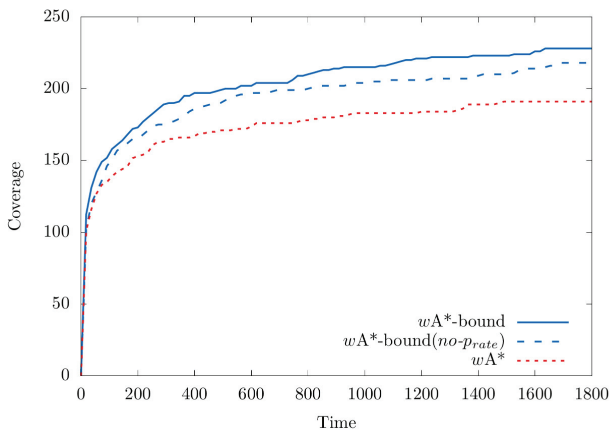

Figure 1: Coverage over time for wA* with h^{Bound} (with and without p_{rate}) and the baseline wA* on all the considered benchmark instances.

more CPU-time is used, and larger areas of the search space are explored, the h^{Bound} obtains better coverage results than the baseline when more than approx. 100 CPU-time seconds are given to solve instances. In terms of coverage, the use of p_{rate} is clearly useful in h^{Bound}, but the benefit is not evenly spread with regards to the given runtime limits. Our intuition is that the usefulness of the p_{rate} for h^{Bound} depends on the domain and on the quality of the predicted cost. Interestingly, the coverage gap between wA*-bound and wA* tend to slightly increase with the increase of the given CPU-time limits. This suggests that the performance gap in favour of the wA*-bound could further increase for CPU-time limits above 1800 seconds.

Conclusions

In this paper we have addressed the problem of exploiting plan cost predictions, computed at preprocessing through machine learning techniques, in order to improve planning performance for propositional domains with action costs.

An effective use of these predictions during planning should take into account that the predicted cost can be (even grossly) inaccurate with respect to the best quality plan that a planning approach can found within a certain time limit. We have developed a cost prediction model that is based on standard machine learning techniques using a large set of instance features, and we have proposed a method to exploit the predictions made by this model in the context of wA*. Our method can be seen as a way to dynamically adjust the input weight w of the heuristic during search, taking into account the predicted plan cost and preventing the search to be misguided when the prediction is severely inaccurate.

As a future work we aim at investigating an extension of our approach to predict and exploit makespan and plan cost in metric-temporal planning (Fox and Long 2003; Gerevini, Saetti, and Serina 2008) and for plan adaptation (Gerevini and Serina 2010).

Acknowledgments. We thank the anonymous reviewers for their useful comments. This work was supported by MIUR

"Fondo Dipartimenti di Eccellenza 2018-2022" of the DII Department at the University of Brescia.

References

Areces, C.; Bustos, F.; Dominguez, M.; and Hoffmann, J. 2014. Optimizing planning domains by automatic action schema splitting. In *Proc. of ICAPS 2014*.

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. 2005. Macro-FF: improving AI planning with automatically learned macro-operators. *J. Artif.Intell. Res. (JAIR)* 24:581–621.

Celorrio, S. J.; de la Rosa, T.; Fernández, S.; Fernández, F.; and Borrajo, D. 2012. A review of machine learning for automated planning. *The Knowledge Engineering Review (KER)* 27(4):433–467.

Cenamor, I., and Pozanco, A. 2019. Insights from the 2018 ipc benchmarks. In *Proc. of the Workshop of the IPC WIPC*.

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2016. The ibacop planning system: Instance-based configured portfolios. *J. Artif.Intell. Res. (JAIR)* 56:657–691.

Chrpa, L.; Vallati, M.; and McCluskey, T. L. 2019. Inner entanglements: Narrowing the search in classical planning by problem reformulation. *Computational Intelligence* 35(2):395–429.

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 *Proc. of ICAPS 2014*.

Fink, E. 1998. How to solve it automatically: Selection among problem-solving methods. In *Proc. of AIPS'98*, 128–136.

Fox, M., and Long, D. 2003. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res. (JAIR)* 20:61–124.

Gerevini, A., and Serina, I. 2010. Efficient plan adaptation through replanning windows and heuristic goals. *Fundam. Inform.* 102(3-4):287–323.

Gerevini, A.; Saetti, A.; and Serina, I. 2006. An approach to temporal planning and scheduling in domains with predictable exogenous events. *J. Artif. Intell. Res.* 25:187–231.

Gerevini, A.; Saetti, A.; and Serina, I. 2008. An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artif. Intell.* 172(8-9):899–944.

Gerevini, A.; Saetti, A.; and Serina, I. 2011. An empirical analysis of some heuristic features for planning through local search and action graphs. *Fundam. Inform.* 107(2-3):167–197.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated planning - theory and practice*. Elsevier.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The weka data mining software: an update. *ACM SIGKDD explorations* 11(1):10–18.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proc. of ICAPS 2009*.

Helmert, M. 2006. The Fast Downward planning system. *J. Artif.Intell. Res. (JAIR)* 26:191–246.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *J. Artif.Intell. Res. (JAIR)* 14:253–302.

Hoffmann, J. 2011. Analyzing search topology without running any search: On the connection between causal graphs and h+. *J. Artif.Intell. Res. (JAIR)* 41:155–229.

Howe, A.; Dahlman, E.; Hansen, C.; Von Mayrhauser, A.; and Scheetz, M. 1999. Exploiting competitive planner performance. In *Proc. of ECP-99*, 62–72.

Kotthoff, L.; Thornton, C.; Hoos, H. H.; Hutter, F.; and Leyton-Brown, K. 2017. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research* 18:25:1–25:5.

Lipovetzky, N., and Geffner, H. 2011. Searching for plans with carefully designed probes. In *Proc. of ICAPS 2011*, 154–161.

Nakhost, H.; Müller, M.; Valenzano, R.; and Xie, F. 2011. Arvand: the art of random walks. In *Booklet of the Seventh International Planning Competition*.

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

Rintanen, J. 2012. Engineering efficient planners with SAT. In *Proc. of ECAI 2012*, 684–689.

Rizzini, M.; Fawcett, C.; Vallati, M.; Gerevini, A. E.; and Hoos, H. H. 2017. Static and dynamic portfolio methods for optimal planning: An empirical analysis. *International Journal on Artificial Intelligence Tools* 26(1):1–27.

Roberts, M.; Howe, A. E.; Wilson, B.; and desJardins, M. 2008. What makes planners predictable? In *Proc. of ICAPS 2008*, 288–295.

Scala, E., and Torasso, P. 2015. Deordering and numeric macro actions for plan repair. In *Proc. of IJCAI 2015*, 1673–1681.

Scala, E. 2014. Plan repair for resource constrained tasks via numeric macro actions. In *Prod. of ICAPS 2014*.

Seipp, J.; Sievers, S.; Helmert, M.; and Hutter, F. 2015. Automatic configuration of sequential planning portfolios. In *Proc. of AAAI 2015*, 3364–3370.

Stern, R.; Felner, A.; van den Berg, J.; Puzis, R.; Shah, R.; and Goldberg, K. 2014. Potential-based bounded-cost search and anytime non-parametric A*. *Artificial Intelligence* 214:1–25.

Thayer, J. T., and Ruml, W. 2011. Bounded suboptimal search: A direct approach using inadmissible estimates. In *Proc. of IJCAI 2011*, 674–679.

Vallati, M., and Serina, I. 2018. A general approach for configuring PDDL problem models. In *Proc. of ICAPS 2019*, 431–436.