Hapori Greedy

Patrick Ferber[1], Michael Katz[2], Jendrik Seipp[3], Silvan Sievers[1], Daniel Borrajo[4], Isabel Cenamor, Tomas de la Rosa, Fernando Fernandez-Rebollo[4], Carlos Linares López[4], Sergio Nuñez, Alberto Pozanco, Horst Samulowitz[2], Shirin Sohrabi[2]

[1] University of Basel, Switzerland
[2] IBM T.J. Watson Research Center, Yorktown Heights, USA
[3] Linköping University, Sweden
[4] Universidad Carlos III de Madrid, Spain

patrick.ferber@unibas.ch, michael.katz1@ibm.com, jendrik.seipp@liu.se, silvan.sievers@unibas.ch, dborrajo@ia.uc3m.es, icenamorg@gmail.com, tomdelarosa@gmail.com, ffernand@inf.uc3m.es, clinares@inf.uc3m.es, sergio.nunez@repsol.com, alberto.pozanco@gmail.com, samulowitz@us.ibm.com, shirin.sohrabi@gmail.com

Abstract

Hapori Greedy[1] is a portfolio planner which participated in the optimal, satisficing, and agile tracks of the International Planning Competition (IPC) 2023. It uses the greedy algorithm by Streeter, Golovin, and Smith (2007) to compute a sequential static portfolio over IPC 2018 planners in an offline preprocessing phase.

Definitions

Before we describe the greedy portfolio computation algorithm, we give some definitions concerning planning tasks, sequential portfolios and quality metrics.

Informally speaking, a classical planning task consists of an initial state, a goal description and a set of operators. In the setting of *satisficing planning*, solving a planning task entails finding any operator sequence that leads from the initial state to a goal state, with a preference for cheap solutions. On the other hand, in the setting of *agile planning*, the task is to find solutions as fast as possible, regardless of the solution cost. The third setting we consider in this planner abstract is *bounded-cost* planning, where plans must not be more expensive than a given bound.

We define $c(A, I, t)$ as the *cost* of the solution a planning algorithm A finds for planning task I within time t, or as ∞ if it does not find a solution in that time. Furthermore, we let $c^\star(I)$ denote the *minimum known solution cost* for task I (approximated by a set of Fast Downward configurations). Following IPC evaluation criteria, we define the *solution quality* $q_{\text{sol}}(A, I, t) = \frac{c^\star(I)}{c(A,I,t)}$ as the minimum known solution cost divided by the solution cost achieved by A in time t.

A *sequential planning portfolio* P is a sequence of pairs $\langle A, t\rangle$ where A is a planning algorithm and $t \in \mathbb{N}_{>0}$ is the time limit in seconds for A. We denote the portfolio resulting from appending a component $\langle A, t\rangle$ to a portfolio P by $P \oplus \langle A, t\rangle$.

We now define two quality scores $q(P, I)$ that evaluate the performance of a portfolio P on task I. In the satisficing and bounded-cost settings we use the solution quality $q_{\text{sol}}(P, I)$.

[1]Hapori is the Maori word for community.

Algorithm 1 Greedy algorithm by Streeter, Golovin, and Smith (2007) computing a sequential portfolio for a given quality function q, algorithms $\mathcal{A}$, instances $\mathcal{I}$ and total portfolio runtime T.

```
1:  function ComputePortfolio($q$, $\mathcal{A}$, $\mathcal{I}$, $T$)
2:      $P \leftarrow \langle\rangle$
3:      $t_{\text{used}} \leftarrow 0$
4:      while $t_{\max} = T - t_{\text{used}} > 0$ do
5:          $\langle A, t\rangle \leftarrow \arg\max_{\langle A', t'\rangle \in \mathcal{A} \times [1, t_{\max}]} q_\Delta(P, A', t', \mathcal{I})$
6:          if $q_\Delta(P, A, t, \mathcal{I}) = 0$ then
7:              return $P$
8:          $P \leftarrow P \oplus \langle A, t\rangle$
9:          $t_{\text{used}} \leftarrow t_{\text{used}} + t$
10:     return $P$
```

It is the maximum solution quality any of the components in P achieves for I, i.e.,

$$q_{\text{sol}}(P, I) = \max_{\langle A,t\rangle \in P} q_{\text{sol}}(A, I, t).$$

Following IPC 2018 evaluation criteria, for the agile planning setting we define *agile quality* as

$$q_{\text{agile}}(P, I) = \begin{cases} 0 & \text{if } t(P, I) > T \\ 1 & \text{if } t(P, I) \le 1 \\ 1 - \frac{\log_{10} t(P,I)}{\log_{10}(T)} & \text{otherwise} \end{cases},$$

where $t(P, I)$ is the time that portfolio P needs to solve task I and T is the total portfolio runtime.

A portfolio's score on multiple tasks $\mathcal{A}$ is defined as the sum of the individual scores, i.e., $q(P, \mathcal{I}) = \sum_{I \in \mathcal{I}} q(P, I)$, and the score of the empty portfolio is always 0.

Greedy Portfolio Computation Algorithm

We now describe the greedy algorithm by Streeter, Golovin, and Smith (2007). Given a quality score q, a set of algorithms $\mathcal{A}$, a set of tasks $\mathcal{I}$ and the total portfolio runtime T, the greedy algorithm iteratively constructs a sequential portfolio.

As shown in Algorithm 1, the procedure starts with an empty portfolio P (line 2) and then iteratively selects an algorithm $A \in \mathcal{A}$ and a time limit $t \in [1, t_{\max}]$ (discretized to seconds) for A such that adding $\langle A, t \rangle$ to P improves P the most (line 5). The quality improvement between P and $P \oplus \langle A, t \rangle$ is measured by the q_Δ function:

$$q_\Delta(P, A, t, \mathcal{I}) = \frac{\sum_{I \in \mathcal{I}} q(P \oplus \langle A, t \rangle, I) - q(P, I)}{t}$$

If appending the pair $\langle A, t \rangle$ to P does not change the portfolio quality anymore, we converged and can terminate (line 6). Otherwise, the pair is appended to P (line 8). This process iterates until the sum of the runtimes in the portfolio components exceeds the maximum porfolio runtime T (line 4).

Components and Training Data

Planners. As the pool of planners for our portfolios to choose from, we used all planners from the IPC 2018. If an IPC 2018 planner was already a portfolio, we used its component planners instead. We only considered each planner once (some portfolios included planners that were also submitted separately and several portfolios included the same planners).

For the optimal track, we had to exclude maplan-1, maplan-2, and MSP because they use CPLEX, and Complementary1 because it generates suboptimal solutions. Furthermore, the FDMS planners and Metis1 were covered by Delfi already. This results in the following list of planners (or their components):

- Complementary2 (Franco, Lelis, and Barley 2018)
- components of DecStar (Gnad, Shleyfman, and Hoffmann 2018)
- components of Delfi (Delfi1 and Delfi2 have the same components; Katz et al., 2018b)
- Metis2 (Sievers and Katz 2018)
- Planning-PDBs (Moraru et al. 2018)
- Scorpion (Seipp 2018b)
- SymBA*1 (IPC 2014; Torralba et al., 2014)
- Symple-1 and Symple-2 (Speck, Geißer, and Mattmüller 2018)

All planners participating in the satisficing track also participated in the agile track (except for Fast Downward Stone Soup 2018), with an identical code base but possibly with different configurations. We thus only have one set of planners but multiple configurations for these two tracks. We had to exclude alien because we could not get it to run, and freelunch-doubly-relaxed, fs-blind and fs-sim because they have a large number of dependencies which results in planner images too large to be included in our pool. Furthermore, IBaCoP-2018 and IBaCoP2-2018 use a large number of planners or portfolios of which newer and stronger versions participated in IPC 2018 as standalone planners, or which we failed to get to run, so we only cover the component planners Jasper, Madagascar, Mercury, and Probe. This results in the following list of planners (or their components):

- Cerberus and Cerberus-gl (Katz 2018)
- components of DecStar (Gnad, Shleyfman, and Hoffmann 2018)
- components of Fast Downward Remix (Seipp 2018a)
- components of Fast Downward Stone Soup 2018 (Seipp and Röger 2018)
- Jasper (IPC 2014; Xie, Müller, and Holte, 2014)
- LAPKT-DUAL-BFWS, LAPKT-POLYNOMIAL-BFWS, LAPKT-DFS+, and LAPKT-BFWS-Preference (Francès et al. 2018)
- Madagascar (IPC 2014; Rintanen, 2014)
- Mercury2014 (Katz and Hoffmann 2014)
- MERWIN (Katz et al. 2018a)
- OLCFF (Fickert and Hoffmann 2018)
- Probe (IPC 2014; Lipovetzky et al., 2014)
- Grey Planning configuration of Saarplan (Fickert et al., 2018; rest covered by DecStar)
- Symple-1 and Symple-2 (Speck, Geißer, and Mattmüller 2018)

Benchmarks and Runtime. For training the portfolios, we used all tasks and domains from previous IPCs, from Delfi (Katz et al. 2018b), and from the 21.11 Autoscale collection Torralba, Seipp, and Sievers (2021), leading to a set of 92 domains with 7330 tasks. We used Downward Lab (Seipp et al. 2017) to run all planners on all benchmarks on AMD EPYC 7742 2.25GHz processors, imposing a memory limit of 8 GiB and a time limit of 30 minutes for optimal planners and 5 minutes for satisficing and agile planners. For each run, we stored its outcome (plan found, out of memory, out of time, task not supported by planner, error), the execution time, the maximum resident memory, and if the run found a plan, the plan length and plan cost. This data set is online available.[2] As training data for our optimal (respectively satisficing/agile) portfolios, we selected from each domain the 30 tasks which are solved by the fewest optimal (or satisficing/agile) planners, which results in 1926 (optimal) and 2377 (satisficing/agile) remaining tasks.

Resulting Portfolios

Passing the algorithms and benchmarks described above to the greedy portfolio computation algorithm, together with the quality score q_{sol} and time limit T=1800 seconds, we obtain a portfolio for the optimal track that consists of 14 component planners, all unique, using time limits between 6s and 703s. For the satisficing track, there are 70 component planners, 50 of which are unique (the greedy algorithm often adds the same planner configuration multiple times with different time limits), using time limits between 1s and 267s. For the agile track, there are 45 component planners, 29 of which are unique, using time limits between 1 and 251 seconds.

On the training set with 1926 tasks, the optimal portfolio solves 1663 tasks while the best planner only solves 1258 tasks. The satisficing portfolio trained on 2377 tasks

[2]URL to be published

achieves an overall quality score of 2106.9 compared to 1319.3 by the best planner. The agile portfolio achieves an overall agile score of 1400.3 compared to 1270.6 by the best planner.

Executing Sequential Portfolios

In the previous sections, we assumed that a portfolio simply assigns a runtime to each algorithm, leaving their sequential order unspecified. With the simplifying assumption that all planners use the full assigned time and do not communicate with each other, the order is indeed irrelevant. In reality the situation is more complex since we do not know upfront how long a selected planner will really run. Therefore, we treat per-algorithm time limits defined by the portfolio as relative, rather than absolute values: whenever we start an algorithm, we compute the total allotted time of this and all following algorithms and scale it to the actually remaining computation time. We then assign the respective scaled time to the run. As a result, the last algorithm is allowed to use all of the remaining time.

In the satisficing setting we would like to use the cost of a plan found by one algorithm to prune the search of subsequent planner runs (in the agile setting we stop after finding the first valid plan). However, since we use the planners as black boxes, this is impossible in our setting.

We use the driver component of Fast Downward (Helmert 2006) which implements the above described mechanic for running portfolios.

Acknowledgments

The success of a portfolio planner must be primarily attributed to the developers of the portfolio components. Therefore, we would like to express our gratitude to the numerous authors of the components on which our portfolios are based.

References

Fickert, M.; Gnad, D.; Speicher, P.; and Hoffmann, J. 2018. SaarPlan: Combining Saarland's Greatest Planning Techniques. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 11–16.

Fickert, M.; and Hoffmann, J. 2018. OLCFF: Online-Learning h^{CFF}. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 17–19.

Francès, G.; Geffner, H.; Lipovetzky, N.; and Ramiréz, M. 2018. Best-First Width Search in the IPC 2018: Complete, Simulated, and Polynomial Variants. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 23–27.

Franco, S.; Lelis, L. H. S.; and Barley, M. 2018. The Complementary2 Planner in the IPC 2018. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 32–36.

Gnad, D.; Shleyfman, A.; and Hoffmann, J. 2018. DecStar – STAR-topology DECoupled Search at its best. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 42–46.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Katz, M. 2018. Cerberus: Red-Black Heuristic for Planning Tasks with Conditional Effects Meets Novelty Heuristic and Enchanced Mutex Detection. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 47–51.

Katz, M.; and Hoffmann, J. 2014. Mercury Planner: Pushing the Limits of Partial Delete Relaxation. In *Eighth International Planning Competition (IPC-8): Planner Abstracts*, 43–47.

Katz, M.; Lipovetzky, N.; Moshkovich, D.; and Tuisov, A. 2018a. MERWIN Planner: Mercury Enchanced With Novelty Heuristic. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 53–56.

Katz, M.; Sohrabi, S.; Samulowitz, H.; and Sievers, S. 2018b. Delfi: Online Planner Selection for Cost-Optimal Planning. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 57–64.

Lipovetzky, N.; Ramirez, M.; Muise, C.; and Geffner, H. 2014. Width and Inference Based Planners: SIW, BFS(f), and PROBE. In *Eighth International Planning Competition (IPC-8): Planner Abstracts*, 6–7.

Moraru, I.; Edelkamp, S.; Martinez, M.; and Franco, S. 2018. Planning-PDBs Planner. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 69–73.

Rintanen, J. 2014. Madagascar: Scalable Planning with SAT. In *Eighth International Planning Competition (IPC-8): Planner Abstracts*, 66–70.

Seipp, J. 2018a. Fast Downward Remix. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 74–76.

Seipp, J. 2018b. Fast Downward Scorpion. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 77–79.

Seipp, J.; Pommerening, F.; Sievers, S.; and Helmert, M. 2017. Downward Lab. https://doi.org/10.5281/zenodo.790461.

Seipp, J.; and Röger, G. 2018. Fast Downward Stone Soup 2018. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 80–82.

Sievers, S.; and Katz, M. 2018. Metis 2018. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 83–84.

Speck, D.; Geißer, F.; and Mattmüller, R. 2018. SYMPLE: Symbolic Planning based on EVMDDs. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 91–94.

Streeter, M. J.; Golovin, D.; and Smith, S. F. 2007. Combining Multiple Heuristics Online. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007)*, 1197–1203. AAAI Press.

Torralba, Á.; Alcázar, V.; Borrajo, D.; Kissmann, P.; and Edelkamp, S. 2014. SymBA*: A Symbolic Bidirectional A* Planner. In *Eighth International Planning Competition (IPC-8): Planner Abstracts*, 105–109.

Torralba, Á.; Seipp, J.; and Sievers, S. 2021. Automatic Instance Generation for Classical Planning. In Goldman, R. P.;

Biundo, S.; and Katz, M., eds., *Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)*, 376–384. AAAI Press.

Xie, F.; Müller, M.; and Holte, R. 2014. Jasper: the art of exploration in Greedy Best First Search. In *Eighth International Planning Competition (IPC-8): Planner Abstracts*, 39–42.