Fast Downward Cedalion

Jendrik Seipp and **Silvan Sievers**
Universität Basel
Basel, Switzerland
{jendrik.seipp,silvan.sievers}@unibas.ch

Frank Hutter
Universität Freiburg
Freiburg, Germany
fh@informatik.uni-freiburg.de

To avoid duplication of content we only give a high-level overview of our algorithm here and refer to a technical report for details on our methodology (Seipp, Sievers, and Hutter 2013). An extended version of that report is forthcoming. The paper at hand focuses mostly on the configuration setup we used for generating portfolios for the IPC 2014 learning track.

Portfolio Configuration

Cedalion is our algorithm for automatically configuring sequential planning portfolios. Given a parametrized planner and a set of training instances, it iteratively selects the pair of planner configuration and time slice that improves the current portfolio the most per time spent. At the end of each iteration all instances for which the current portfolio finds the best solution are removed from the training set. The algorithm stops when the the total runtime of the added configurations reaches the portfolio time limit (usually 30 minutes) or if the training set becomes empty.

Conceptually, Cedalion is similar to Hydra (Xu, Hoos, and Leyton-Brown 2010) in that both use an algorithm configuration procedure (Hutter 2009) to add the most improving configuration to the existing portfolio in each iteration. However, Hydra uses the algorithm selection system SATzilla (Xu et al. 2008) to select a configuration based on the characteristics of a given test instance, and therefore does not have a notion of time slices. In contrast, Cedalion runs all found configurations sequentially regardless of the instance at hand and makes the length of the time slices part of the configuration space.

Cedalion is also very similar to the greedy algorithm presented by Streeter, Golovin, and Smith (2007). Given a finite set of solvers and their runtimes on the training set, that algorithm iteratively adds the (solver, time slice) pair that most improves the current portfolio per time spent. In contrast, Cedalion does not rely on a priori runtime data and supports infinite sets of solver configurations by using an algorithm configuration procedure to adaptively gather this data for promising configurations only.

In principle, Cedalion could employ any algorithm configuration procedure to select the next (configuration, time slice) pair. Here, we use the model-based configurator SMAC (Hutter, Hoos, and Leyton-Brown 2011) for this task. As a simple standard parallelization method (Hutter, Hoos, and Leyton-Brown 2012), we performed 5 SMAC runs in parallel in every iteration of Cedalion and used the best of the 5 identified (configuration, time slice) pairs.

We could have included planners other than Fast Downward in our Cedalion portfolios (even other parameterized planning frameworks, by configuring on the union of all parameter spaces). If portfolios prove to be useful in the learning track setting, this would have almost certainly improved performance, due to the fact that portfolios can exploit the complementary strengths of diverse approaches. Nevertheless, we chose to limit ourselves to Fast Downward in order to quantify the performance gain possible within this framework.

Optimization Function

Formally, Cedalion uses the following metric m_P to evaluate $\langle \text{configuration}, \text{time slice} \rangle$ pairs $\langle \boldsymbol{\theta}, t \rangle$ on task π given the current Portfolio P:

$$m_P(\langle \boldsymbol{\theta}, t \rangle, \pi) = \frac{q(P \oplus \langle \boldsymbol{\theta}, t \rangle, \pi) - q(P, \pi)}{t}, \qquad (1)$$

where $P \oplus \langle \boldsymbol{\theta}, t \rangle$ denotes the portfolio P with $\langle \boldsymbol{\theta}, t \rangle$ appended and q is a performance metric. Following IPC evaluation criteria, we define $q(P, \pi)$ as the solution quality achieved by portfolio P for task π, i.e., as the minimum known cost for task π divided by the cost achieved by P.

Configuration Setup

Our Fast Downward Cedalion portfolios are the result of using Cedalion to find sequential portfolios of Fast Downward (Helmert 2006) configurations. In this section we describe how we used Cedalion to find portfolios for the IPC 2014 learning track.

Benchmarks

The competition organizers chose 6 benchmark domains and provided problem generators, parameter distributions and the corresponding sample problems for each of them. We chose to use the provided instances as our test set and generated our own training instances by running the problem generators for the same parameter distributions again. Since the generators are nondeterministic our training and test sets (probably) do not intersect. For the domains for which only very few parameter sets for the instance generator were provided (floortile, nomystery, parking and spanner), we generated some additional problems for our training sets (using domain knowledge to choose parameter sets giving rise to somewhat easier instances).

Configuration Space

The set of Fast Downward configurations Cedalion could choose from was the same as the one used by Fawcett et al. (2011), the only exception being that we also included an implementation of the YAHSP lookahead strategy (Vidal 2004). We used a time budget of 60 hours on 5 machines for every iteration of our portfolio construction process (running 5 parallel independent SMAC runs). Cedalion learned between 2 and 6 configurations for each domain. We list the selected configurations and their timeouts in the appendix.

Metrics

For each domain we let Cedalion configure two portfolios: one Cedalion run always added the pair of configuration and time slice to the current portfolio that maximized the additional quality score per time spent, the other ignored the solution quality and instead optimized for coverage, i. e. additionally solved problems per time spent.

For each domain we compared the corresponding two portfolios on the respective test set and found that the portfolios that optimized for quality directly also achieved a higher score for this metric on the test set. (The only domain in which the coverage-optimized portfolio achieved a higher score was floortile.) Since the quality score is also the main metric in the IPC 2014 learning track, we only submitted portfolios that optimized for quality.

Comparison to Fast Downward SMAC

To gain a baseline for Cedalion's performance in the learning setting, we also evaluated Fast Downward SMAC (Seipp, Sievers, and Hutter 2014) which only uses SMAC to find a single good configuration for each domain. On the instances provided by the competition organizers (our test set) we found that each of the two approaches performed better in some cases. In nomystery and spanner Fast Downward Cedalion and Fast Downward SMAC obtained the same quality score (neither approach solved any of the 6 spanner test instances). SMAC fared better than Cedalion for floortile and parking while the opposite was true for elevators and transport. We offer two possible explanations why SMAC by itself could perform better than Cedalion in two domains. On the one hand, the single SMAC runs were more powerful than the SMAC runs performed inside Cedalion: Cedalion could not use adaptive capping (terminating slow runs early (Hutter et al. 2009)) and also used SMAC runs of only 60 hours in each of its iterations. On the other hand, it may be the case that in the relatively homogeneous domains of the learning track, not much can be gained by a portfolio over a single parameter configuration.

Acknowledgments

First, we would like to thank all Fast Downward contributors. Portfolios crucially rely on their base algorithms, and as such a strong portfolio is ultimately due to the work of the developers of these base algorithms. We therefore wish to thank Malte Helmert, Jörg Hoffmann, Erez Karpas, Emil Keyder, Raz Nissim, Florian Pommerening, Silvia Richter, Gabriele Röger and Matthias Westphal for their contributions to the Fast Downward codebase.

Our thanks also go to Manuel Heusner, for allowing us to include his implementation of the YAHSP lookahead strategy in the Cedalion portfolios. As of yet this code is not merged into the main Fast Downward repository.

References

Fawcett, C.; Helmert, M.; Hoos, H.; Karpas, E.; Röger, G.; and Seipp, J. 2011. FD-Autotune: Domain-specific configuration using Fast Downward. In *ICAPS 2011 Workshop on Planning and Learning*, 13–17.

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

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

Hutter, F.; Hoos, H.; and Leyton-Brown, K. 2011. Sequential model-based optimization for general algorithm configuration. In Coello, C. A. C., ed., *Proceedings of the Fifth Conference on Learning and Intelligent OptimizatioN (LION 2011)*, 507–523. Springer.

Hutter, F.; Hoos, H.; and Leyton-Brown, K. 2012. Parallel algorithm configuration. In *Proceedings of the Sixth Conference on Learning and Intelligent OptimizatioN (LION 2012)*, 55–70. Springer.

Hutter, F. 2009. *Automated Configuration of Algorithms for Solving Hard Computational Problems*. Ph.D. Dissertation, University of British Columbia.

Seipp, J.; Sievers, S.; and Hutter, F. 2013. Automatic configuration of sequential planning portfolios. Technical Report CS-2013-005, Universität Basel, Department of Mathematics and Computer Science.

Seipp, J.; Sievers, S.; and Hutter, F. 2014. Fast Downward SMAC. In *Eighth International Planning Competition (IPC-8) Planning and Learning Part: planner abstracts*.

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.

Vidal, V. 2004. A lookahead strategy for heuristic search planning. In Zilberstein, S.; Koehler, J.; and Koenig, S., eds., *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, 150–159. AAAI Press.

Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2008. SATzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research* 32:565–606.

Xu, L.; Hoos, H. H.; and Leyton-Brown, K. 2010. Hydra: Automatically configuring algorithms for portfolio-based selection. In Fox, M., and Poole, D., eds., *Proceedings* *of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, 210–216. AAAI Press.

Appendix – Fast Downward Cedalion Portfolios

We list the configurations found during the configuration processes of our Fast Downward Cedalion portfolios. Our portfolio components have the form of pairs (time slice, configuration), with the first entry reflecting the time slice allowed for the configuration, which is in turn shown in the second component. Note that if the summed up time slices of the configurations of a portfolio is below the overall time limit, the time slices will be stretched at runtime to match the allowed maximal time, i. e. every time slice is multiplied with the same factor such that the portfolio runs for exactly the overall time limit.

- Elevators:

```
22,
--heuristic hCea=cea(cost_type=1)
--search ehc(hCea,preferred=[hCea],preferred_usage=1,cost_type=1)

142,
--landmarks lmg=lm_rhw(reasonable_orders=false,only_causal_landmarks=false,
                       disjunctive_landmarks=false,conjunctive_landmarks=true,
                       no_orders=true,lm_cost_type=0,cost_type=1)
--heuristic hGoalCount=goalcount(cost_type=2)
--heuristic hLM,hFF=lm_ff_syn(lmg,admissible=false)
--heuristic hBlind=blind()
--search
lazy(alt([single(hBlind),single(hBlind,pref_only=true),single(hLM),
          single(hLM,pref_only=true),single(hFF),single(hFF,pref_only=true),
          single(hGoalCount),single(hGoalCount,pref_only=true)],
         boost=4419),preferred=[hLM,hGoalCount],reopen_closed=true,cost_type=1)

35,
--heuristic hCea=cea(cost_type=1)
--search ehc(hCea,preferred=[hCea],preferred_usage=0,cost_type=1)

120,
--heuristic hCea=cea(cost_type=1)
--search lazy(alt([single(sum([g(),weight(hCea,10)])),
                   single(sum([g(),weight(hCea,10)]),pref_only=true)],
                  boost=2285), preferred=[hCea],reopen_closed=false,cost_type=0)

142,
--landmarks lmg=lm_zg(reasonable_orders=false,only_causal_landmarks=false,
                      disjunctive_landmarks=true,conjunctive_landmarks=true,
                      no_orders=true,lm_cost_type=1,cost_type=2)
--heuristic hLM,hFF=lm_ff_syn(lmg,admissible=false)
--heuristic hCg=cg(cost_type=2)
--search lazy(alt([single(sum([g(),weight(hLM,5)])),
                   single(sum([g(),weight(hLM,5)]),pref_only=true),
                   single(sum([g(),weight(hFF,5)])),
                   single(sum([g(),weight(hFF,5)]),pref_only=true),
                   single(sum([g(),weight(hCg,5)])),
                   single(sum([g(),weight(hCg,5)]),pref_only=true)],
                  boost=2395),preferred=[hLM],reopen_closed=true,cost_type=0)
```

- Floortile:

```
1,
--heuristic hBlind=blind()
--heuristic hAdd=add(cost_type=0)
--search lazy(alt([single(sum([weight(g(),4),weight(hBlind,5)])),
                   single(sum([weight(g(),4),weight(hBlind,5)]),pref_only=true),
                   single(sum([weight(g(),4),weight(hAdd,5)])),
                   single(sum([weight(g(),4),weight(hAdd,5)]),pref_only=true)],
                  boost=1627),preferred=[hAdd],reopen_closed=false,cost_type=0)
```

```
1,
--heuristic hFF=ff(cost_type=0)
--heuristic hGoalCount=goalcount(cost_type=0)
--heuristic hAdd=add(cost_type=0)
--heuristic hBlind=blind()
--search lazy(alt([single(sum([g(),hBlind])),
                   single(sum([g(),hBlind]),pref_only=true),
                   single(sum([g(),hFF])),
                   single(sum([g(),hFF]),pref_only=true),
                   single(sum([g(),hAdd])),
                   single(sum([g(),hAdd]),pref_only=true),
                   single(sum([g(),hGoalCount])),
                   single(sum([g(),hGoalCount]),pref_only=true)],
                  boost=702),
              preferred=[hAdd,hGoalCount],reopen_closed=true,cost_type=0)

1,
--landmarks lmg=lm_rhw(reasonable_orders=false,only_causal_landmarks=true,
                       disjunctive_landmarks=false,conjunctive_landmarks=true,
                       no_orders=false,lm_cost_type=0,cost_type=0)
--heuristic hHMax=hmax()
--heuristic hLM,hFF=lm_ff_syn(lmg,admissible=true)
--search lazy(alt([tiebreaking([sum([weight(g(),4),weight(hLM,5)]),hLM]),
                   tiebreaking([sum([weight(g(),4),weight(hLM,5)]),hLM],
                               pref_only=true),
                   tiebreaking([sum([weight(g(),4),weight(hFF,5)]),hFF]),
                   tiebreaking([sum([weight(g(),4),weight(hFF,5)]),hFF],
                               pref_only=true),
                   tiebreaking([sum([weight(g(),4),weight(hHMax,5)]),hHMax]),
                   tiebreaking([sum([weight(g(),4),weight(hHMax,5)]),hHMax],
                               pref_only=true)],
                  boost=4054),preferred=[hFF],reopen_closed=false,cost_type=0)

52,
--landmarks lmg=lm_merged([lm_rhw(),lm_hm(m=1)],reasonable_orders=false,
                          only_causal_landmarks=true,disjunctive_landmarks=true,
                          conjunctive_landmarks=true,no_orders=true,
                          lm_cost_type=1,cost_type=0)
--heuristic hLM,hFF=lm_ff_syn(lmg,admissible=true)
--heuristic hHMax=hmax()
--heuristic hAdd=add(cost_type=0)
--search lazy(alt([single(sum([g(),hLM])),
                   single(sum([g(),hLM]),pref_only=true),
                   single(sum([g(),hFF])),
                   single(sum([g(),hFF]),pref_only=true),
                   single(sum([g(),hHMax])),
                   single(sum([g(),hHMax]),pref_only=true),
                   single(sum([g(),hAdd])),
                   single(sum([g(),hAdd]),pref_only=true)],
                  boost=4240),
              preferred=[hFF,hAdd],reopen_closed=false,cost_type=0)

207,
--heuristic hAdd=add(cost_type=1)
--heuristic hLMCut=lmcut(cost_type=1)
--heuristic hCea=cea(cost_type=0)
--heuristic hHMax=hmax()
--search eager(alt([tiebreaking([sum([weight(g(),8),weight(hLMCut,9)]),hLMCut]),
                    tiebreaking([sum([weight(g(),8),weight(hLMCut,9)]),hLMCut],
                                pref_only=true),
                    tiebreaking([sum([weight(g(),8),weight(hHMax,9)]),hHMax]),
                    tiebreaking([sum([weight(g(),8),weight(hHMax,9)]),hHMax],
                                pref_only=true),
                    tiebreaking([sum([weight(g(),8),weight(hAdd,9)]),hAdd]),
                    tiebreaking([sum([weight(g(),8),weight(hAdd,9)]),hAdd],
                                pref_only=true),
                    tiebreaking([sum([weight(g(),8),weight(hCea,9)]),hCea]),
                    tiebreaking([sum([weight(g(),8),weight(hCea,9)]),hCea],
                                pref_only=true)],
                   boost=3409),
               preferred=[hAdd],reopen_closed=true,pathmax=false,cost_type=1)

216,
--landmarks lmg=lm_exhaust(reasonable_orders=false,only_causal_landmarks=false,
                           disjunctive_landmarks=true,conjunctive_landmarks=true,
                           no_orders=false,lm_cost_type=2,cost_type=0)
--heuristic hCea=cea(cost_type=2)
--heuristic hLM,hFF=lm_ff_syn(lmg,admissible=false)
--heuristic hAdd=add(cost_type=0)
--heuristic hBlind=blind()
--search eager(alt([tiebreaking([sum([g(),weight(hBlind,2)]),hBlind]),
                    tiebreaking([sum([g(),weight(hBlind,2)]),hBlind],
                                pref_only=true),
                    tiebreaking([sum([g(),weight(hLM,2)]),hLM]),
                    tiebreaking([sum([g(),weight(hLM,2)]),hLM],pref_only=true),
                    tiebreaking([sum([g(),weight(hFF,2)]),hFF]),
                    tiebreaking([sum([g(),weight(hFF,2)]),hFF],pref_only=true),
                    tiebreaking([sum([g(),weight(hAdd,2)]),hAdd]),
                    tiebreaking([sum([g(),weight(hAdd,2)]),hAdd],pref_only=true),
                    tiebreaking([sum([g(),weight(hCea,2)]),hCea]),
                    tiebreaking([sum([g(),weight(hCea,2)]),hCea],pref_only=true)],
                   boost=949),
               preferred=[hFF],reopen_closed=false,pathmax=true,cost_type=1)
```

- Nomystery:

```
1,
--landmarks lmg=lm_hm(reasonable_orders=false,only_causal_landmarks=false,
                      disjunctive_landmarks=true,conjunctive_landmarks=true,
                      no_orders=false,m=1,cost_type=2)
--heuristic hLM=lmcount(lmg,admissible=false,pref=false,cost_type=2)
--heuristic hCg=cg(cost_type=2)
--search lazy(alt([single(sum([weight(g(),4),weight(hLM,5)])),
                   single(sum([weight(g(),4),weight(hCg,5)]))],
                  boost=4741),preferred=[],reopen_closed=true,cost_type=0)

1,
--landmarks lmg=lm_rhw(reasonable_orders=true,only_causal_landmarks=true,
                       disjunctive_landmarks=true,conjunctive_landmarks=true,
                       no_orders=false,cost_type=1)
--heuristic hLM=lmcount(lmg,admissible=false,pref=false,cost_type=1)
--heuristic hGoalCount=goalcount(cost_type=2)
--search eager(alt([single(sum([weight(g(),4),weight(hLM,5)])),
                    single(sum([weight(g(),4),weight(hLM,5)]),pref_only=true),
                    single(sum([weight(g(),4),weight(hGoalCount,5)])),
                    single(sum([weight(g(),4),weight(hGoalCount,5)]),
                           pref_only=true)],
                    boost=1146),
                preferred=[hGoalCount],reopen_closed=true,pathmax=true,cost_type=0)

3,
--heuristic hFF=ff(cost_type=2)
--search lazy(alt([tiebreaking([sum([weight(g(),4),weight(hFF,5)]),hFF]),
                   tiebreaking([sum([weight(g(),4),weight(hFF,5)]),hFF],
                               pref_only=true)],
                  boost=1410),preferred=[hFF],reopen_closed=false,cost_type=0)

1,
--landmarks lmg=lm_merged([lm_rhw(),lm_hm(m=1)],reasonable_orders=false,
                          only_causal_landmarks=false,disjunctive_landmarks=true,
                          conjunctive_landmarks=true,no_orders=false,cost_type=0)
--heuristic hHMax=hmax()
--heuristic hLM=lmcount(lmg,admissible=true,pref=false,cost_type=0)
--heuristic hAdd=add(cost_type=0)
--search lazy(alt([tiebreaking([sum([weight(g(),2),weight(hLM,3)]),hLM]),
                   tiebreaking([sum([weight(g(),2),weight(hLM,3)]),hLM],
                               pref_only=true),
                   tiebreaking([sum([weight(g(),2),weight(hHMax,3)]),hHMax]),
                   tiebreaking([sum([weight(g(),2),weight(hHMax,3)]),hHMax],
                               pref_only=true),
                   tiebreaking([sum([weight(g(),2),weight(hAdd,3)]),hAdd]),
                   tiebreaking([sum([weight(g(),2),weight(hAdd,3)]),hAdd],
                               pref_only=true)],
                  boost=1215),preferred=[hAdd],reopen_closed=true,cost_type=0)
```

- Parking:

```
4,
--heuristic hFF=ff(cost_type=1)
--heuristic hBlind=blind()
--heuristic hGoalCount=goalcount(cost_type=0)
--search lazy(alt([tiebreaking([sum([g(),weight(hBlind,3)]),hBlind]),
                   tiebreaking([sum([g(),weight(hBlind,3)]),hBlind],
                               pref_only=true),
                   tiebreaking([sum([g(),weight(hFF,3)]),hFF]),
                   tiebreaking([sum([g(),weight(hFF,3)]),hFF],pref_only=true),
                   tiebreaking([sum([g(),weight(hGoalCount,3)]),hGoalCount]),
                   tiebreaking([sum([g(),weight(hGoalCount,3)]),hGoalCount],
                               pref_only=true)],
                  boost=487),
              preferred=[hFF,hGoalCount],reopen_closed=false,cost_type=0)

28,
--heuristic hFF=ff(cost_type=1)
--heuristic hCea=cea(cost_type=2)
--heuristic hGoalCount=goalcount(cost_type=1)
--search eager(alt([single(hFF),single(hFF,pref_only=true),single(hCea),
                    single(hCea,pref_only=true),single(hGoalCount),
                    single(hGoalCount,pref_only=true)],boost=2337),
               preferred=[hGoalCount],reopen_closed=false,pathmax=true,
               lookahead=true,la_greedy=true,la_repair=false,cost_type=0)

27,
--heuristic hCea=cea(cost_type=1)
--heuristic hAdd=add(cost_type=2)
--heuristic hGoalCount=goalcount(cost_type=0)
```

```
--search lazy(alt([single(sum([g(),weight(hAdd,5)])),
                   single(sum([g(),weight(hAdd,5)]),pref_only=true),
                   single(sum([g(),weight(hCea,5)])),
                   single(sum([g(),weight(hCea,5)]),pref_only=true),
                   single(sum([g(),weight(hGoalCount,5)])),
                   single(sum([g(),weight(hGoalCount,5)]),pref_only=true)],
                  boost=151),preferred=[hAdd,hCea],reopen_closed=true,cost_type=0)

30,
--landmarks lmg=lm_zg(reasonable_orders=false,only_causal_landmarks=false,
                      disjunctive_landmarks=true,conjunctive_landmarks=true,
                      no_orders=true,cost_type=1)
--heuristic hCg=cg(cost_type=1)
--heuristic hCea=cea(cost_type=0)
--heuristic hLMCut=lmcut(cost_type=2)
--heuristic hLM=lmcount(lmg,admissible=false,pref=false,cost_type=1)
--heuristic hFF=ff(cost_type=1)
--search lazy(alt([single(sum([g(),hFF])),
                   single(sum([g(),hFF]),pref_only=true),
                   single(sum([g(),hLM])),
                   single(sum([g(),hLM]),pref_only=true),
                   single(sum([g(),hLMCut])),
                   single(sum([g(),hLMCut]),pref_only=true),
                   single(sum([g(),hCg])),
                   single(sum([g(),hCg]),pref_only=true),
                   single(sum([g(),hCea])),
                   single(sum([g(),hCea]),pref_only=true)],
                  boost=1986),
              preferred=[hFF,hCg,hCea],reopen_closed=false,cost_type=0)

328,
--heuristic hGoalCount=goalcount(cost_type=1)
--heuristic hFF=ff(cost_type=1)
--heuristic hCg=cg(cost_type=0)
--heuristic hAdd=add(cost_type=2)
--search lazy(alt([tiebreaking([sum([g(),weight(hFF,2)]),hFF]),
                   tiebreaking([sum([g(),weight(hFF,2)]),hFF],pref_only=true),
                   tiebreaking([sum([g(),weight(hAdd,2)]),hAdd]),
                   tiebreaking([sum([g(),weight(hAdd,2)]),hAdd],pref_only=true),
                   tiebreaking([sum([g(),weight(hCg,2)]),hCg]),
                   tiebreaking([sum([g(),weight(hCg,2)]),hCg],pref_only=true),
                   tiebreaking([sum([g(),weight(hGoalCount,2)]),hGoalCount]),
                   tiebreaking([sum([g(),weight(hGoalCount,2)]),hGoalCount],
                               pref_only=true)],
                  boost=506),preferred=[hFF,hAdd],reopen_closed=true,cost_type=1)

66,
--landmarks lmg=lm_exhaust(reasonable_orders=false,only_causal_landmarks=false,
                           disjunctive_landmarks=true,conjunctive_landmarks=true,
                           no_orders=true,cost_type=2)
--heuristic hCg=cg(cost_type=1)
--heuristic hLM=lmcount(lmg,admissible=true,pref=false,cost_type=2)
--heuristic hBlind=blind()
--heuristic hGoalCount=goalcount(cost_type=1)
--heuristic hCea=cea(cost_type=0)
--heuristic hFF=ff(cost_type=1)
--search lazy(alt([single(sum([g(),weight(hBlind,5)])),
                   single(sum([g(),weight(hBlind,5)]),pref_only=true),
                   single(sum([g(),weight(hFF,5)])),
                    single(sum([g(),weight(hFF,5)]),pref_only=true),
                    single(sum([g(),weight(hLM,5)])),
                    single(sum([g(),weight(hLM,5)]),pref_only=true),
                    single(sum([g(),weight(hCg,5)])),
                    single(sum([g(),weight(hCg,5)]),pref_only=true),
                    single(sum([g(),weight(hCea,5)])),
                    single(sum([g(),weight(hCea,5)]),pref_only=true),
                    single(sum([g(),weight(hGoalCount,5)])),
                    single(sum([g(),weight(hGoalCount,5)]),pref_only=true)],
                   boost=2886),
              preferred=[hFF,hCea],reopen_closed=false,cost_type=1)
```

- Spanner:

```
10,
--heuristic hBlind=blind()
--heuristic hAdd=add(cost_type=1)
--heuristic hGoalCount=goalcount(cost_type=2)
--heuristic hFF=ff(cost_type=1)
--search eager(alt([tiebreaking([sum([weight(g(),2),weight(hBlind,3)]),hBlind]),
                    tiebreaking([sum([weight(g(),2),weight(hFF,3)]),hFF]),
                    tiebreaking([sum([weight(g(),2),weight(hAdd,3)]),hAdd]),
                    tiebreaking([sum([weight(g(),2),weight(hGoalCount,3)]),
                                 hGoalCount])],
                   boost=537),
               preferred=[],reopen_closed=true,pathmax=false,lookahead=true,
               la_greedy=true,la_repair=false,cost_type=0)

187,
--heuristic hBlind=blind()
--heuristic hFF=ff(cost_type=2)
--search eager(alt([single(sum([g(),weight(hBlind,3)])),
                    single(sum([g(),weight(hFF,3)]))],
                   boost=3764),
               preferred=[],reopen_closed=true,pathmax=true,lookahead=true,
               la_greedy=true,la_repair=false,cost_type=0)
```

- Transport:

```
11,
--heuristic hBlind=blind()
--heuristic hFF=ff(cost_type=0)
--heuristic hCg=cg(cost_type=0)
--search eager(alt([tiebreaking([sum([g(),weight(hBlind,7)]),hBlind]),
                    tiebreaking([sum([g(),weight(hBlind,7)]),hBlind],
                                pref_only=true),
                    tiebreaking([sum([g(),weight(hFF,7)]),hFF]),
                    tiebreaking([sum([g(),weight(hFF,7)]),hFF],pref_only=true),
                    tiebreaking([sum([g(),weight(hCg,7)]),hCg]),
                    tiebreaking([sum([g(),weight(hCg,7)]),hCg],pref_only=true)],
                   boost=4960),
               preferred=[hCg],reopen_closed=false,pathmax=false,lookahead=true,
               la_greedy=true,la_repair=false,cost_type=0)

21,
--heuristic hGoalCount=goalcount(cost_type=1)
--heuristic hCg=cg(cost_type=2)
--heuristic hFF=ff(cost_type=2)
--heuristic hBlind=blind()
```

```
--search eager(alt([single(sum([g(),weight(hBlind,2)])),
                    single(sum([g(),weight(hBlind,2)]),pref_only=true),
                    single(sum([g(),weight(hFF,2)])),
                    single(sum([g(),weight(hFF,2)]),pref_only=true),
                    single(sum([g(),weight(hCg,2)])),
                    single(sum([g(),weight(hCg,2)]),pref_only=true),
                    single(sum([g(),weight(hGoalCount,2)])),
                    single(sum([g(),weight(hGoalCount,2)]),pref_only=true)],
                   boost=3134),
               preferred=[hCg],reopen_closed=false,pathmax=false,
               lookahead=true,la_greedy=true,la_repair=true,cost_type=0)

48,
--landmarks lmg=lm_rhw(reasonable_orders=true,only_causal_landmarks=false,
                       disjunctive_landmarks=false,conjunctive_landmarks=true,
                       no_orders=true,lm_cost_type=2,cost_type=0)
--heuristic hCg=cg(cost_type=1)
--heuristic hLM,hFF=lm_ff_syn(lmg,admissible=false)
--search lazy(alt([single(hLM),
                   single(hLM,pref_only=true),
                   single(hFF),
                   single(hFF,pref_only=true),
                   single(hCg),
                   single(hCg,pref_only=true)],
                  boost=4527),preferred=[hLM],reopen_closed=false,cost_type=1)

249,
--landmarks lmg=lm_zg(reasonable_orders=false,only_causal_landmarks=false,
                      disjunctive_landmarks=true,conjunctive_landmarks=true,
                      no_orders=false,lm_cost_type=1,cost_type=2)
--heuristic hLM,hFF=lm_ff_syn(lmg,admissible=false)
--heuristic hCg=cg(cost_type=0)
--search lazy(alt([tiebreaking([sum([g(),weight(hLM,10)]),hLM]),
                   tiebreaking([sum([g(),weight(hLM,10)]),hLM],pref_only=true),
                   tiebreaking([sum([g(),weight(hFF,10)]),hFF]),
                   tiebreaking([sum([g(),weight(hFF,10)]),hFF],pref_only=true),
                   tiebreaking([sum([g(),weight(hCg,10)]),hCg]),
                   tiebreaking([sum([g(),weight(hCg,10)]),hCg],pref_only=true)],
                  boost=3655),preferred=[hLM,hCg],reopen_closed=true,cost_type=1)

250,
--landmarks lmg=lm_merged([lm_rhw(),lm_hm(m=1)],reasonable_orders=true,
                          only_causal_landmarks=true,disjunctive_landmarks=true,
                          conjunctive_landmarks=true,no_orders=false,cost_type=1)
--heuristic hBlind=blind)
--heuristic hLM=lmcount(lmg,admissible=false,pref=false,cost_type=1)
--heuristic hGoalCount=goalcount(cost_type=0)
--heuristic hCea=cea(cost_type=1)
--search lazy(alt([tiebreaking([sum([g(),weight(hBlind,10)]),hBlind]),
                   tiebreaking([sum([g(),weight(hBlind,10)]),hBlind],
                               pref_only=true),
                   tiebreaking([sum([g(),weight(hLM,10)]),hLM]),
                   tiebreaking([sum([g(),weight(hLM,10)]),hLM],pref_only=true),
                   tiebreaking([sum([g(),weight(hCea,10)]),hCea]),
                   tiebreaking([sum([g(),weight(hCea,10)]),hCea],pref_only=true),
                   tiebreaking([sum([g(),weight(hGoalCount,10)]),hGoalCount]),
                   tiebreaking([sum([g(),weight(hGoalCount,10)]),hGoalCount],
                               pref_only=true)],
                  boost=1807),preferred=[hCea],reopen_closed=true,cost_type=1)
```

```
71,
--heuristic hCg=cg(cost_type=0)
--heuristic hAdd=add(cost_type=0)
--search lazy(alt([tiebreaking([sum([g(),weight(hAdd,10)]),hAdd]),
                   tiebreaking([sum([g(),weight(hAdd,10)]),hAdd],pref_only=true),
                   tiebreaking([sum([g(),weight(hCg,10)]),hCg]),
                   tiebreaking([sum([g(),weight(hCg,10)]),hCg],pref_only=true)],
                  boost=3885),preferred=[hAdd],reopen_closed=true,cost_type=0)
```