Delfi: Online Planner Selection for Cost-Optimal Planning

Michael Katz and **Shirin Sohrabi** and **Horst Samulowitz**
IBM Research
Yorktown Heights, NY, USA
michael.katz1@ibm.com, ssohrab@us.ibm.com, samulowitz@us.ibm.com

Silvan Sievers
University of Basel
Basel, Switzerland
silvan.sievers@unibas.ch

Abstract

Cost-optimal planning has not seen many successful approaches that work well across all domains. Some cost-optimal planners excel on some domains, while exhibiting less exciting performance on others. For a particular domain, however, there is often a cost-optimal planner that works extremely well. For that reason, portfolio-based techniques have recently become popular. These either decide offline on a particular resource allocation scheme for a given collection of planners or try to perform an online classification of a given planning task to select a planner to be applied to solving the task at hand.

Our planner *Delfi* is an online portfolio planner. In contrast to existing techniques, Delfi exploits deep learning techniques to learn a model that predicts which of the planners in the portfolio can solve a given planning task within the imposed time and memory bounds. Delfi uses graphical representations of a planning task which allows exploiting existing tools for image convolution. In this planner abstract, we describe the techniques used to create our portfolio planner.

Introduction

As planning is known to be computationally hard even for extremely conservative problem formalisms (Bylander 1994), no single planner should be expected to work well on all planning domains, or even on all tasks in a particular domain. As a result, research has not only focused on developing different planning techniques, such as improving search or heuristics, but also on exploiting multiple diverse approaches for solving planning tasks.

One such a approach is to aggregate multiple planners in a portfolio (Seipp et al. 2012; Vallati 2012; Cenamor, de la Rosa, and Fernández 2013; Seipp et al. 2015), which is what we do in this work. Such portfolios are often *sequential* and defined by two decisions: (i) which planner of the available to run next, and (ii) for how long to run it until the next planner is selected. Furthermore, the portfolio-based approaches can be partitioned in those that make those decisions ahead of time, called *offline* portfolios (Helmert et al. 2011; Núñez, Borrajo, and Linares López 2014; Seipp, Sievers, and Hutter 2014a; 2014b; 2014c) and those that make these decisions per given input task, called *online* portfolios (Cenamor, de la Rosa, and Fernández 2014).

Our planner, called *Delfi* for *DEap Learning of PortFoIIos*, is an online portfolio planner submitted to optimal classical track of the International Planning Competition (IPC) 2018. It consists of (a) a collection of cost-optimal planners based on Fast Downward (Helmert 2006), and (b) a module that, given a planning task, selects the planner from the collection for which the confidence that it solves the given planning task is highest. Once selected, the planner is run on the given task for the entire available time. In the remainder of this planner abstract, we describe both components in detail.

Collection of Cost-Optimal Planners

The large literature on classical planning results in an extensive pool of available planning systems that we could in principle all use. However, there are a few aspects that guided our decision to collect a rather small subset of specific planners. Firstly, the task of integrating the diverse planners within one system able to run them all in the same setting is a big (technical) challenge, and evaluating all of these planners for the training phase of learning the model would be extremely time-consuming. Secondly, portfolio planners always suffer from clearly identifying their components that are primarily responsible for the good performance of the portfolio planner.

Bearing in mind the first aspect, we restricted the pool of planners to those based on Fast Downward (Helmert 2006). This has the additional advantage that we also exploit how far a portfolio exclusively based on a single planning system fares. With respect to the second aspect, we excluded all recent (and state-of-the-art) planners that have not been evaluated in any previous competition. In particular, many of these planners are submitted independently to the IPC 2018. Furthermore, we mainly focused on planners with main components that we co-developed in order to primarily evaluate our own contributions.

These considerations result in a collection of 17 planners for our portfolio planner Delfi. With the exception of SymBA* (Torralba et al. 2014), the winner of the IPC 2014, included as-is in our collection of planners, all planners are based on a recent version of Fast Downward. These 16 planners use A* search (Hart, Nilsson, and Raphael 1968) and differ in the subsets of the following additional components they use. Please refer to the Appendix for the complete list of planner configurations of our collection, which is identical for both variants of Delfi.

- Pruning based on partial order reduction using strong stubborn sets (Wehrle and Helmert 2014). Delfi uses the implementation of strong stubborn sets available in Fast Downward, which is based on the original implementation of Alkhazraji et al. (2012) and Wehrle and Helmert (2012) that has also been used in Metis 2014 (Alkhazraji et al. 2014). However, the current implementation has been improved in terms of efficiency since its original development.[1] To support conditional effects, we extended the implementation in the same way as in Metis 2014. We also use the same mechanism that disables pruning after the first 1000 expansions if only 10% or fewer states have been pruned at this point. This component is part of all 16 planners.
- Pruning based on structural symmetries (Shleyfman et al. 2015) using DKS (Domshlak, Katz, and Shleyfman 2012) or orbit space search (OSS) (Domshlak, Katz, and Shleyfman 2015). We extended the original implementation of problem description graphs, also called symmetry graphs, which serve as basis for computing symmetries, to support conditional effects. Sievers et al. (2017) recently formally defined this extension in the context of structural symmetries of lifted representations. Out of the 16 planners, 8 use DKS search and the other 8 use OSS, without any other further difference except that merge-and-shrink configurations with OSS need to disable pruning of unreachable states to avoid incorrectly reporting pruned states as dead ends (cf. Sievers et al., 2015, for more details).
- Admissible heuristics:
  - The blind heuristic.
  - The LM-cut heuristic (Helmert and Domshlak 2009). To support conditional effects, we implemented a variant of the LM-cut heuristic that considers effect conditions in the same way as Metis 2014 (Alkhazraji et al. 2014) does. However, we refrain from choosing the regular LM-cut heuristic or the variant that supports conditional effects depending on the requirements of the input planning task, and instead always use the latter implementation that comes with a small overhead due to the need for different data structures.
  - The canonical pattern database (CPDB) heuristic with hillclimbing (HC) to compute pattern collections, also referred to as iPDB in the literature (Haslum et al. 2007). We add a time limit of 900s to the hillclimbing algorithm and denote the planner by HC-CPDB.
  - The zero-one cost partitioning pattern database (ZOPDB) heuristic with a genetic algorithm (GA) to compute pattern collections (Edelkamp 2006). We call the planner GA-ZOPDB.
  - Four variants of the merge-and-shrink heuristic (Dräger, Finkbeiner, and Podelski 2009; Helmert et al. 2014; Sievers 2017). Three of them use the state-of-the-art shrink strategy based on bisimulation (Nissim, Hoffmann, and Helmert 2011) with a size limit of 50000 states on transition systems, always allowing (perfect) shrinking, called B. The fourth variant uses a greedy variant of B, called G, not imposing any size limit on transition systems, and also always allowing shrinking. All configurations use full pruning (Sievers 2017), i.e., always prune both unreachable and irrelevant states, unless combined with OSS as discussed above, in which case pruning of unreachable states is disabled. We perform exact label reductions based on Θ-combinability (Sievers, Wehrle, and Helmert 2014) with a fixed point algorithm using a random order on factors.

    Finally, all variants use a time limit of 900s for computing the heuristic, which leads to computing so-called *partial* merge-and-shrink abstractions that do not cover all variables of the task whenever the time limit is hit. In these cases, we pick one of the remaining induced heuristics according to the following rule of thumb: we prefer the heuristic with the largest estimate for the initial state (rationale: better informed heuristic), breaking ties in favor of larger factors (rationale: more fine-grained abstraction), and choose a random heuristic among all remaining candidates of equal preference. For more details on this, we refer to the paper introducing partial abstractions (Sievers 2018b) and the separate competition entry called Fast Downward Merge-and-Shrink (Sievers 2018a) which uses the same merge-and-shrink configurations as our portfolio.

    The remaining difference between the four variants is the merge strategy, which finally results in the following merge-and-shrink configurations:
    * B-SCCdfp: the state-of-the-art merge strategy based on *strongly connected components* of the causal graph (Sievers, Wehrle, and Helmert 2016), which uses DFP (Sievers, Wehrle, and Helmert 2014) for internal merging.
    * B-MIASMdfp: the entirely precomputed merge strategy *maximum intermediate abstraction size minimizing* (Fan, Müller, and Holte 2014), which uses DFP as a fallback mechanism.
    * B-sbMIASM (previously also called DYN-MIASM): the merge strategy *score-based MIASM* (Sievers, Wehrle, and Helmert 2016), which is a simple variant of MIASM.
    * G-SCCdfp: as SCCdfp, but with the greedy variant of bisimulation-based shrinking.

  As mentioned above, each heuristic is used in two planners, once with OSS and once with DKS. For the two PDB-based heuristics that do not support conditional effects natively, we compile away conditional effects by multiplying out all operators, adding copies for each possible scenario of different subsets of satisfied effect conditions and operator preconditions.
- Postprocessing the SAS$^+$ representation obtained with the translator of Fast Downward (Helmert 2009) by using the implementation of h^2 mutex detection of Alcázar and Torralba (2015). This component is present in 14

[1]See `http://issues.fast-downward.org/issue499` and `http://issues.fast-downward.org/issue628`.

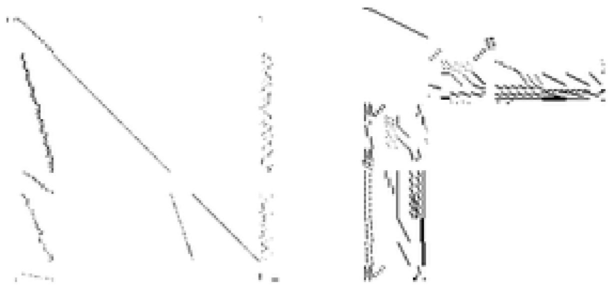

Figure 1: Images constructed from lifted and grounded representations of task pfile01-001.pddl of BARMAN-OPT11.

out of 16 planners. The two planners that do not exploit h^2 mutexes use the merge-and-shrink configuration B-MIASMdfp (once with OSS, once with DKS) which heavily relies on remaining mutexes in the SAS^+ representation. Our preliminary experiments showed that using the postprocessing in this case significantly harmed the performance.

Online Planner Selection

The online planner selection of Delfi is based on a model that predicts for all planners of the portfolio whether they solve a given planning task within the fixed resource and time limits of the competition or not. To learn such a model, we created a collection of tasks to serve as training set, ran all planners in our collection on these tasks to find whether they solve the task or not, and used the resulting data to train a deep neural network. In what follows, we describe how we created the data as well as how we trained the model.

Data Creation

Our collection of tasks includes all benchmarks of the classical tracks of all IPCs as well as some domains from the learning tracks. We further include the domains BRIEFCASEWORLD, FERRY, and HANOI from the IPP benchmark collection (Köhler 1999), and the genome edit distance (GEDP) domain (Haslum 2011). We also use domains generated by the conformant-to-classical planning compilation (T0) (Palacios and Geffner 2009) and the finite-state controller synthesis compilation (FSC) (Bonet, Palacios, and Geffner 2009). In addition to existing tasks of these domains, we generated additional ones for some domains where generators were available. Please see the Appendix for a complete list of used domains. To filter out too hard tasks, we removed all tasks from the training set that were not solved by any of our planners.

Data Representation

To be able to take advantage of existing deep learning tools, we need to represent planning tasks in a way that can be consumed by these tools. In the context of solving other model-based problems, such as SAT and CSP, Loreggia et al. (2016) converted the textual description of input problems to a grayscale image by converting each character to a

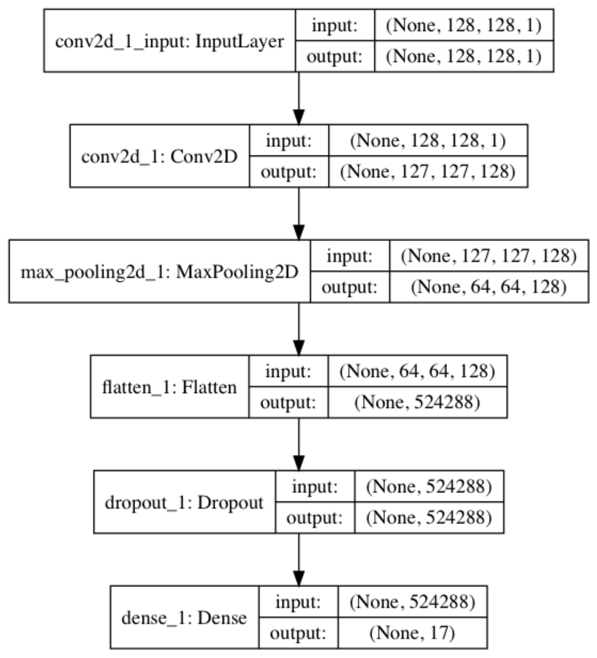
conv2d_1_input: InputLayer
input: (None, 128, 128, 1)
output: (None, 128, 128, 1)
conv2d_1: Conv2D
input: (None, 128, 128, 1)
output: (None, 127, 127, 128)
max_pooling2d_1: MaxPooling2D
input: (None, 127, 127, 128)
output: (None, 64, 64, 128)
flatten_1: Flatten
input: (None, 64, 64, 128)
output: (None, 524288)
dropout_1: Dropout
input: (None, 524288)
output: (None, 524288)
dense_1: Dense
input: (None, 524288)
output: (None, 17)

Figure 2: Visualization of the model graph structure.

pixel. Inspired by their ideas, we also chose to represent each task by a grayscale image of a constant size of $128 * 128$ pixels.

However, in contrast to Loreggia et al. (2016), we chose to abstract from the textual representation and decided to use a structural representation of planning tasks, namely the *abstract structure* (Sievers et al. 2017), which encodes the PDDL description of the task. We either directly convert this abstract structure to a graph by computing the abstract structure graph as described by Sievers et al. (2017), or we first ground the abstract structure (which corresponds to grounding the planning task) and the turn it into a graph. In the latter case, we technically do not use the abstract structure graph but the conceptually equivalent *problem description graph* (Shleyfman et al. 2015), which usually is used to compute symmetries of a ground task. Finally, we turn the graph into a grayscale image that represents the adjacency matrix of the graph and reduce the grayscale image to the desired constant size.

In some preliminary experiments, we experimented with both ways of creating an image for a planning task and decided to use both. Delfi 1 computes the image from the lifted representation of the task, and Delfi 2 from the grounded one. (This is the only difference of the two variants of our planner.) Figure 1 illustrates the different images we obtain for a task of the BARMAN domain.

Model Creation

Our tool of choice for both model creation and training is Keras (Chollet 2015) with Tensorflow as a backend. For our model, we employ a simple convolutional neural network (CNN) (LeCun, Bengio, and Hinton 2015) consisting of one convolutional layer, one pooling layer, one dropout layer, and one hidden layer. The main reason to choose a network with few parameters is to reduce the chances of overfitting given the comparably limited amount of data we created. Figure 2 shows the structure of the CNN.

We model planner performance by a binary feature that indicates whether the planner solves a task within the given time (1800 seconds) and memory (7744 MiB) limits or not. We also experimented with using the actual runtime, hence not predicting whether a planner solves a task or not, but rather predicting the runtime of the planner on the task. However, our preliminary tests indicated that the performance of the network when using the binary feature is comparable to when using the actual runtime. Our conjecture is that this is due to the relatively small amount of training data and due to the fact that the model learned with the binary feature bases the decision for a planner on the confidence that this planner solves the task, which means that it is likely to prefer faster planners. As a result, we decided to use the simpler representation in our model.

Consequently, we trained the CNN by optimizing for binary cross-entropy (Rubinstein 1997) so that each planner has a certain probability assigned to it that indicates how likely it is to solve a problem within the limits. Although our CNN is rather simple, it still features a range of model hyper-parameters, which we fine-tuned employing the approach by Diaz et al. (2017).[2] For both lifted and grounded representations, the hyper-parameter optimization found very similar parameters and thus we choose the same parameters in both cases, which are as follows. The convolutional layer filter size is 3, the pooling filter size is 1, and the dropout rate is 0.48. The CNN is optimized using Stochastic Gradient Decent with learning rate 0.1, decay 0.04, momentum 0.95, nesterov set to FALSE and a batch size of 52.

Post-IPC Analysis

In the following, we evaluate the performance of the two Delfi planners in the optimal classical track of the IPC 2018. While Delfi 2 finished 7 among all 16 submissions, Delfi 1 took the first place. To assess the contribution of the individual components of the portfolios, we ran them on all benchmarks under IPC conditions. We report both performance on the training set on which we learned prior to the IPC and performance on the new planning benchmarks from the competition, which we will refer to as the test set. In addition to the individual results, we include the competition results of Delfi 1 and 2[3] and of the oracle planner that takes the maximum over all planners on a per-instance base. Finally, as a baseline for portfolio performance on the test set, we also evaluate the uniform portfolio that runs each planner from the portfolio with a equal time share of the IPC limit of 30 minutes.

Table 1 shows aggregated coverage of the training set. For full domain-wise coverage on the training set, see Table 4 in the Appendix. We see that both variants of Delfi (coverage of 2282 and 2236) greatly improve over the best single planner in our portfolio, LM-cut (coverage of 1956). At the same time, the oracle portfolio solves 2350 tasks, which gives rise to the hope that the learned models of the Delfi planners are not overfitted too much on the training set.

Table 2 shows domain-wise coverage of the test set, using the same way as the IPC to compute aggregated results for the two domains where two formulations have been used (CALDERA and ORGANIC-SYNTHESIS), which is to take the task-wise maximum performance of each planner. Looking at the performance of the individual planners, we find that their coverage of different domains is diverse. In particular, Symba and HC-PDB (iPDB) are very complementary, but also LM-cut and some merge-and-shrink variants achieve best coverage in some domains.

Looking at the performance of the portfolios, we see that the uniform portfolio does not improve over the best individual planner, Symba, but even solves fewer tasks. While Delfi 1 is much stronger than the baseline uniform portfolio, the same is not true for Delfi 2, which even lacks behind the best individual planner Symba. We discuss the difference between Delfi 1 and 2 in more detail below. Finally, it is also worth pointing out that the oracle planner solves 18 tasks out of 240 more than Delfi 1, which leads us to conclude that the learned model of Delfi 1 generalized very well to the test set that the IPC 2018 benchmarks represent.

We now analyze the Delfi planners in more detail. Table 3 shows the number of times a component planner was chosen by Delfi: the first block shows the number of times a specific planner was chosen for each domain, and the second block shows the number of times each planner was chosen in total. We consider both variants of the two domains with two different formulations (CALDERA and ORGANIC-SYNTHESIS) individually rather than the combined domain, since it is not clear which planner to consider the chosen one if a different planner was chosen for both formulations.[4]

Delfi 1 often consistently chooses one or few planners in a given domain (with the exception of NURIKABE), which seems to be reasonable since we expect the same planner to be strong for different tasks across a given domain. Delfi 2, on the other hand, more frequently chooses a larger variety of planners in a domain.[5] While we cannot explain this significant difference yet, it is clearly due to the difference of the representation of planning tasks, i.e., the difference

[2]To evaluate our approach prior to the competition, we held back the IPC 2014 domains as a separate validation set. Only when learning the final model for the competition, we used the full benchmark set with the previously determined fine-tuned hyperparameters.

[3]We also re-ran both Delfi planners ourselves but decided to stick with the official competition results. The differences in coverage and planner selection per domain were marginal.

[4]However, for our set of planners, we note that we could restrict the analysis to the original formulation of CALDERA, in which OSS-LM-cut solves 13 tasks like Delfi 1, and to ORGANIC-SYNTHESIS-SPLIT, which dominates ORGANIC-SYNTHESIS for all planners.

[5]In 13 ORGANIC-SYNTHESIS tasks, the translator runs out of memory so that Delfi 2 cannot compute the problem description graph. In 3 ORGANIC-SYNTHESIS-SPLIT and 2 NURIBAKE tasks, the translator hits the time limit of 60s that we imposed for graph computation. In all of cases, Delfi 2 cannot use its model for planner selection but uses the fallback planner DSK-lmc.

	Portfolio Components																	Delfi		Orcl
	DKS								OSS									1	2	
	blind	lmc	P1	P2	M1	M2	M3	M4	blind	lmc	P1	P2	M1	M2	M3	M4	Sym			
C (3721)	1470	1956	1836	1602	1796	1645	1767	1615	1472	1948	1838	1606	1782	1643	1707	1568	1867	2282	2236	**2350**

Table 1: Coverage of the training set. Abbreviations: lmc: LM-cut; P1: HC-PDB; P2: GA-ZOPDB; M1: B-SCCdfp; M2: B-MIASMdfp; M3: B-sbMIASM; M4: G-SCCdfp; Sym: SymBA* 2014; Orcl: oracle portfolio over all component planners.

	Portfolio Components																	Unif	Delfi		Orcl
	DKS								OSS										1	2	
	blind	lmc	P1	P2	M1	M2	M3	M4	blind	lmc	P1	P2	M1	M2	M3	M4	Sym				
agricola (20)	5	0	7	5	6	0	10	5	6	0	7	6	6	0	6	6	13	7	12	11	**14**
caldera-comb (20)	12	13	**16**	13	12	0	12	12	12	13	**16**	13	12	0	12	12	12	13	13	11	**16**
data-network (20)	6	12	11	9	10	10	9	3	6	12	11	9	10	10	9	3	**13**	**13**	**13**	**13**	**13**
nurikabe (20)	10	**12**	**12**	11	11	**12**	**12**	11	10	**12**	**12**	11	11	11	11	11	11	10	**12**	11	**12**
org-syn-comb (20)	**14**	**14**	13	**14**	13	7	13	13	**14**	**14**	13	**14**	13	7	13	13	**14**	13	13	13	**14**
petri-net-al (20)	2	9	0	2	2	0	2	2	2	9	0	2	2	0	2	2	**20**	15	**20**	9	**20**
settlers (20)	8	**9**	0	0	**9**	**9**	8	**9**	8	**9**	0	0	**9**	**9**	8	**9**	**9**	6	**9**	8	**9**
snake (20)	11	7	**14**	11	11	7	11	10	11	7	**14**	11	11	6	11	10	4	10	11	7	**14**
spider (20)	11	11	**14**	11	11	0	11	3	11	11	**14**	11	11	0	11	3	7	13	11	7	**14**
termes (20)	7	6	12	10	10	10	11	6	7	6	12	10	10	10	11	6	**18**	13	12	15	**18**
Sum (200)	86	93	99	86	95	55	99	74	87	93	99	87	95	53	94	75	121	113	126	105	**144**

Table 2: Coverage of the test set (IPC 2018 benchmarks). Abbreviations: lmc: LM-cut; P1: HC-PDB; P2: GA-ZOPDB; M1: B-SCCdfp; M2: B-MIASMdfp; M3: B-sbMIASM; M4: G-SCCdfp; Sym: SymBA* 2014; Unif: uniform portfolio over all component planners; Orcl: oracle portfolio over all component planners.

	Delfi 1	Delfi 2
agricola (20)	Sym (20)	Sym (19), OSS-P1 (1)
caldera (20)	OSS-lmc (20)	Sym (10), DKS-M2 (3), DKS-M1 (2), DKS-lmc (2) DKS-P1 (1), DKS-M3 (1), OSS-lmc (1)
caldera-split (20)	Sym (7), OSS-lmc (8), DKS-lmc (5)	Sym (6), OSS-lmc (6), DKS-lmc (4), DKS-M4 (2), OSS-P1 (1), DKS-M1 (1)
data-network (20)	Sym (17), OSS-lmc (3)	Sym (16), OSS-M2 (3), OSS-P1 (1)
nurikabe (20)	Sym (6), DKS-P1 (6), DKS-blind (2), OSS-P1 (2), DKS-lmc (2), DKS-M2 (1), OSS-lmc (1)	Sym(12), DKS-lmc (3), OSS-M2 (2), DKS-M2 (1), time out (2)
organic-synthesis (20)	Sym (18), OSS-lmc (2)	OSS-P1 (3), Sym(2), DKS-P1 (1), OSS-M2 (1), memory out (13)
organic-synthesis-split (20)	Sym (20)	Sym (14), OSS-lmc (2), DKS-P1 (1), time out (3)
petri-net-alignment (20)	Sym (20)	DKS-lmc (10), OSS-lmc (9), Sym(1)
settlers (20)	OSS-lmc (20)	Sym (20)
snake (20)	OSS-M1 (9), OSS-P1 (7), DKS-P1 (2), Sym (2)	DKS-lmc (12), Sym (8)
spider (20)	DKS-M3 (6), DKS-M1 (6), OSS-M1 (5), DKS-lmc (3)	Sym (16), DKS-lmc (3), OSS-lmc (1)
termes (20)	DKS-M2 (20)	Sym (10), DKS-lmc (6), OSS-M2 (4)
Sum	240	222 (13 memory out, 5 time out)
Symba	110	134
OSS-lmc	54	19
DKS-lmc	10	40
OSS-P1	9	6
DKS-P1	8	3
OSS-M1	14	0
DKS-M1	6	3
DKS-M2	21	4
OSS-M2	0	3
DKS-M3	6	1
DKS-blind	2	0

Table 3: Top: domain-wise number of tasks a planner is selected by our portfolios. Bottom: for each planner, number of times it is selected by our portfolios in total.

between the abstract structure graph (Delfi 1) and the problem description graph (Delfi 2). One important distinguishing feature of this difference is that the problem description graph represents the grounded task (SAS^+), which is a specific representation that depends on the used grounding and invariant synthesis algorithms, while the abstract structure graph represents the lifted representation (PDDL) of the task.

In the benchmark set, there is an observable difference that may explain the different choices to some extent: the IPC 2018 domains exhibit much more conditional effects than the domains of our training set. When we performed initial experiments on the reduced training set (excluding the IPC 2014 domains as a validation set), we observed that using the problem description graph (Delfi 2) resulted in a stronger performance than using the abstract structure graph (Delfi 1) due to better choices of suitable planners. The same is not true anymore when looking at the full training set performance and the test set performance reported here. In future work, we plan to further investigate the differences in the representation of planning tasks and their impact on planner selection.

To assess how well Delfi selects planners for a given task, we also investigate which planners are required to achieve oracle performance. Surprisingly, we found two set covers of only size 3 that cover all tasks solved by any planner: the first consists of Symba, DKS-M3, and DKS-P1, and the second one of Symba, DKS-M3, and OSS-P1. In particular, it is enough to consider Symba, one variant of PDB-based planners, and one variant of merge-and-shrink-based planners (for NURIBAKE), and there is no need for LM-cut or any of the other heuristics at all. While both variants of Delfi frequently choose Symba and sometimes choose PDB-based and merge-and-shrink-based planners, they choose LM-cut the second most of times, even though this would not be required. A possible reason is that LM-cut performs much better on the training set than PDB-based and merge-and-shrink-based planners and hence is more likely to be chosen also on the test set.

Conclusions

In this planner abstract, we described the Delfi planners that participated in the optimal classical track of the IPC 2018. The Delfi planners are online portfolios that select a component planner deemed suitable for the given task based on a model learned with deep learning techniques. In particular, to represent planning tasks, we used two graphical representations for planning tasks, turned them into images and used a convolutional neural network to learn a model that predicts whether a planner solves a given task or not. The performance of Delfi 1, taking the first place in the competition, showed that the learned model generalized well to the new benchmarks used in the competition.

In future work, we would like to greatly extend the set of component planners of our portfolio to evaluate how far the performance of Delfi can be pushed if using many available state-of-the-art planners. We also plan to investigate alternative learning techniques that operate directly on the graph representations rather than relying on images created from these graphs.

Acknowledgments

Since any portfolio crucially depends on the quality of its component planners, we want to thank all contributors of Fast Downward, the authors of Symba 2014, and the authors of the h^2-based preprocessor.

References

Alcázar, V., and Torralba, Á. 2015. A reminder about the importance of computing and exploiting invariants in planning. In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 2–6. AAAI Press.

Alkhazraji, Y.; Wehrle, M.; Mattmüller, R.; and Helmert, M. 2012. A stubborn set algorithm for optimal planning. In De Raedt, L.; Bessiere, C.; Dubois, D.; Doherty, P.; Frasconi, P.; Heintz, F.; and Lucas, P., eds., *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, 891–892. IOS Press.

Alkhazraji, Y.; Katz, M.; Mattmüller, R.; Pommerening, F.; Shleyfman, A.; and Wehrle, M. 2014. Metis: Arming Fast Downward with pruning and incremental computation. In *Eighth International Planning Competition (IPC-8): planner abstracts*, 88–92.

Bonet, B.; Palacios, H.; and Geffner, H. 2009. Automatic derivation of memoryless policies and finite-state controllers using classical planners. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 34–41. AAAI Press.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69(1–2):165–204.

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2013. Learning predictive models to configure planning portfolios. In *ICAPS 2013 Workshop on Planning and Learning*, 14–22.

Cenamor, I.; de la Rosa, T.; and Fernández, F. 2014. IBaCoP and IBaCoPB planner. In *Eighth International Planning Competition (IPC-8): planner abstracts*, 35–38.

Chollet, F. 2015. `https://keras.io`.

Diaz, G. I.; Fokoue-Nkoutche, A.; Nannicini, G.; and Samulowitz, H. 2017. An effective algorithm for hyperparameter optimization of neural networks. *IBM Journal of Research and Development* 61(4).

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced symmetry breaking in cost-optimal planning as forward search. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 343–347. AAAI Press.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2015. Symmetry breaking in deterministic planning as forward search:

Orbit space search algorithm. Technical Report IS/IE-2015-03, Technion.

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2009. Directed model checking with distance-preserving abstractions. *International Journal on Software Tools for Technology Transfer* 11(1):27–37.

Edelkamp, S. 2006. Automated creation of pattern database search heuristics. In Edelkamp, S., and Lomuscio, A., eds., *Proceedings of the 4th Workshop on Model Checking and Artificial Intelligence (MoChArt 2006)*, 35–50.

Fan, G.; Müller, M.; and Holte, R. 2014. Non-linear merging strategies for merge-and-shrink based on variable interactions. In Edelkamp, S., and Barták, R., eds., *Proceedings of the Seventh Annual Symposium on Combinatorial Search (SoCS 2014)*, 53–61. AAAI Press.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI 2007)*, 1007–1012. AAAI Press.

Haslum, P. 2011. Computing genome edit distances using domain-independent planning. In *ICAPS 2011 Scheduling and Planning Applications woRKshop*, 45–51.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 162–169. AAAI Press.

Helmert, M.; Röger, G.; Seipp, J.; Karpas, E.; Hoffmann, J.; Keyder, E.; Nissim, R.; Richter, S.; and Westphal, M. 2011. Fast Downward Stone Soup. In *IPC 2011 planner abstracts*, 38–45.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-shrink abstraction: A method for generating lower bounds in factored state spaces. *Journal of the ACM* 61(3):16:1–63.

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

Helmert, M. 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence* 173:503–535.

Köhler, J. 1999. Handling of conditional effects and negative goals in IPP. Technical Report 128, University of Freiburg, Department of Computer Science.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(7553):436–444.

Loreggia, A.; Malitsky, Y.; Samulowitz, H.; and Saraswat, V. A. 2016. Deep learning for algorithm portfolios. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, 1280–1286. AAAI Press.

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011. Computing perfect heuristics in polynomial time: On bisimulation and merge-and-shrink abstraction in optimal planning. In Walsh, T., ed., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 1983–1990. AAAI Press.

Núñez, S.; Borrajo, D.; and Linares López, C. 2014. Miplan and dpmplan. In *Eighth International Planning Competition (IPC-8): planner abstracts*.

Palacios, H., and Geffner, H. 2009. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research* 35:623–675.

Rubinstein, R. Y. 1997. Optimization of computer simulation models with rare events. *European Journal of Operational Research* 99(1):89–112.

Seipp, J.; Braun, M.; Garimort, J.; and Helmert, M. 2012. Learning portfolios of automatically tuned planners. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 368–372. AAAI Press.

Seipp, J.; Sievers, S.; Helmert, M.; and Hutter, F. 2015. Automatic configuration of sequential planning portfolios. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, 3364–3370. AAAI Press.

Seipp, J.; Sievers, S.; and Hutter, F. 2014a. Fast Downward Cedalion. In *Eighth International Planning Competition (IPC-8): planner abstracts*, 17–27.

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

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

Shleyfman, A.; Katz, M.; Helmert, M.; Sievers, S.; and Wehrle, M. 2015. Heuristics and symmetries in classical planning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, 3371–3377. AAAI Press.

Sievers, S.; Wehrle, M.; Helmert, M.; and Katz, M. 2015. An empirical case study on symmetry handling in cost-optimal planning as heuristic search. In Hölldobler, S.; Krötzsch, M.; Peñaloza-Nyssen, R.; and Rudolph, S., eds., *Proceedings of the 38th Annual German Conference on Artificial Intelligence (KI 2015)*, volume 9324 of *Lecture Notes in Artificial Intelligence*, 151–165. Springer-Verlag.

Sievers, S.; Röger, G.; Wehrle, M.; and Katz, M. 2017. Structural symmetries of the lifted representation of classical planning tasks. In *ICAPS 2017 Workshop on Heuristics and Search for Domain-independent Planning (HSDIP)*, 67–74.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized label reduction for merge-and-shrink heuristics. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, 2358–2366. AAAI Press.

Sievers, S.; Wehrle, M.; and Helmert, M. 2016. An analysis of merge strategies for merge-and-shrink heuristics. In

Coles, A.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 294–298. AAAI Press.

Sievers, S. 2017. *Merge-and-shrink Abstractions for Classical Planning: Theory, Strategies, and Implementation*. Ph.D. Dissertation, University of Basel.

Sievers, S. 2018a. Fast Downward merge-and-shrink. In *Ninth International Planning Competition (IPC-9): planner abstracts*, 77–81.

Sievers, S. 2018b. Merge-and-shrink heuristics for classical planning: Efficient implementation and partial abstractions. In *Proceedings of the 11th Annual Symposium on Combinatorial Search (SoCS 2018)*, 90–98. 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.

Vallati, M. 2012. A guide to portfolio-based planning. In Sombattheera, C.; Loi, N. K.; Wankar, R.; and Quan, T. T., eds., *Proceedings of the 6th International Workshop on Multi-disciplinary Trends in Artificial Intelligence (MIWAI 2012)*, volume 7694, 57–68. Springer.

Wehrle, M., and Helmert, M. 2012. About partial order reduction in planning and computer aided verification. In McCluskey, L.; Williams, B.; Silva, J. R.; and Bonet, B., eds., *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS 2012)*, 297–305. AAAI Press.

Wehrle, M., and Helmert, M. 2014. Efficient stubborn sets: Generalized algorithms and selection strategies. In Chien, S.; Fern, A.; Ruml, W.; and Do, M., eds., *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*, 323–331. AAAI Press.

Appendix

Collection of Planner Configurations

The following are the configurations for the 16 Fast Downward based planners.

1.
```
--symmetries ’sym=structural_symmetries(search_symmetries=dks)’
--search ’astar(blind,symmetries=sym,
    pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)’
```

2.
```
--symmetries ’sym=structural_symmetries(search_symmetries=dks)’
--search ’astar(celmcut,symmetries=sym,
    pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)’
```

3.
```
--symmetries ’sym=structural_symmetries(search_symmetries=dks)’
--search ’astar(
    merge_and_shrink(shrink_strategy=shrink_bisimulation(greedy=false),
        merge_strategy=merge_sccs(order_of_sccs=topological,merge_selector=score_based_filtering(scoring_functions=
        [goal_relevance,dfp,total_order(atomic_before_product=false,atomic_ts_order=reverse_level,product_ts_order=
        new_to_old)])),label_reduction=exact(before_shrinking=true,before_merging=false),max_states=50000,
        threshold_before_merge=1,max_time=900),
    symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)’
```

4.
```
--symmetries ’sym=structural_symmetries(search_symmetries=dks)’
--search ’astar(
    merge_and_shrink(shrink_strategy=shrink_bisimulation(greedy=false),
        merge_strategy=merge_stateless(merge_selector=score_based_filtering(scoring_functions=[sf_miasm(
        shrink_strategy=shrink_bisimulation,max_states=50000),total_order(atomic_before_product=true,
        atomic_ts_order=reverse_level,product_ts_order=old_to_new)])),label_reduction=exact(before_shrinking=true,
        before_merging=false),max_states=50000,threshold_before_merge=1,max_time=900),
    symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)’
```

5.
```
--symmetries ’sym=structural_symmetries(search_symmetries=dks)’
--search ’astar(
    merge_and_shrink(shrink_strategy=shrink_bisimulation(greedy=false),merge_strategy=merge_precomputed(
        merge_tree=miasm(abstraction=miasm_merge_and_shrink(),fallback_merge_selector=score_based_filtering(
        scoring_functions=[goal_relevance,dfp,total_order(atomic_ts_order=reverse_level,product_ts_order=
        new_to_old,atomic_before_product=false)]))),label_reduction=exact(before_shrinking=true,before_merging=false),
        max_states=50000,threshold_before_merge=1,max_time=900),
    symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)’
```

6.
```
--symmetries ’sym=structural_symmetries(search_symmetries=dks)’
--search ’astar(
    merge_and_shrink(shrink_strategy=shrink_bisimulation(greedy=true),merge_strategy=merge_sccs(order_of_sccs=
        topological, merge_selector=score_based_filtering(scoring_functions=[goal_relevance,dfp,
        total_order(atomic_before_product=false, atomic_ts_order=level,product_ts_order=random)])),
        label_reduction=exact(before_shrinking=true,before_merging=false),
        max_states=infinity,threshold_before_merge=1,max_time=900),
    symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)’
```

7.
```
--symmetries ’sym=structural_symmetries(search_symmetries=dks)’
--search ’astar(cpdbs(patterns=hillclimbing(max_time=900),transform=multiply_out_conditional_effects),
    symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)’
```

8.
```
--symmetries ’sym=structural_symmetries(search_symmetries=dks)’
--search ’astar(zopdbs(patterns=genetic(pdb_max_size=50000,num_collections=5,num_episodes=30,
    mutation_probability=0.01), transform=multiply_out_conditional_effects),symmetries=sym,
    pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01), num_por_probes=1000)’
```

9.
```
--symmetries ’sym=structural_symmetries(search_symmetries=oss)’
--search ’astar(blind,symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)’
```

10.
```
--symmetries ’sym=structural_symmetries(search_symmetries=oss)’
--search ’astar(celmcut,symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)’
```

11.
```
--symmetries ’sym=structural_symmetries(search_symmetries=oss)’
--search ’astar(
    merge_and_shrink(shrink_strategy=shrink_bisimulation(greedy=false),merge_strategy=merge_sccs(order_of_sccs=
        topological, merge_selector=score_based_filtering(scoring_functions=[goal_relevance,dfp,
        total_order(atomic_before_product=false, atomic_ts_order=reverse_level,product_ts_order=new_to_old)])),
        label_reduction=exact(before_shrinking=true, before_merging=false),max_states=50000,threshold_before_merge=1,
        max_time=900,prune_unreachable_states=false),
    symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)’
```

12.
```
--symmetries 'sym=structural_symmetries(search_symmetries=oss)'
--search 'astar(
    merge_and_shrink(shrink_strategy=shrink_bisimulation(greedy=false),merge_strategy=merge_stateless(merge_selector=
        score_based_filtering(scoring_functions=[sf_miasm(shrink_strategy=shrink_bisimulation,max_states=50000),
        total_order(atomic_before_product=true,atomic_ts_order=reverse_level,product_ts_order=old_to_new)])),
        label_reduction=exact(before_shrinking=true,before_merging=false),
        max_states=50000,threshold_before_merge=1,max_time=900,prune_unreachable_states=false),
    symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)'
```

13.
```
--symmetries 'sym=structural_symmetries(search_symmetries=oss)'
--search 'astar(
    merge_and_shrink(shrink_strategy=shrink_bisimulation(greedy=false),merge_strategy=merge_precomputed(merge_tree=
        miasm(abstraction=miasm_merge_and_shrink(),fallback_merge_selector=score_based_filtering(scoring_functions=
        [goal_relevance,dfp,total_order(atomic_ts_order=reverse_level,product_ts_order=new_to_old,
        atomic_before_product=false)]))),label_reduction=exact(before_shrinking=true,before_merging=false),
        max_states=50000,threshold_before_merge=1,max_time=900,prune_unreachable_states=false),
    symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)'
```

14.
```
--symmetries 'sym=structural_symmetries(search_symmetries=oss)'
--search 'astar(
    merge_and_shrink(shrink_strategy=shrink_bisimulation(greedy=true),merge_strategy=merge_sccs(order_of_sccs=
        topological, merge_selector=score_based_filtering(scoring_functions=[goal_relevance,dfp,
        total_order(atomic_before_product=false,atomic_ts_order=level,product_ts_order=random)])),
        label_reduction=exact(before_shrinking=true, efore_merging=false),max_states=infinity,
        threshold_before_merge=1,max_time=900,prune_unreachable_states=false),
    symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)'
```

15.
```
--symmetries 'sym=structural_symmetries(search_symmetries=oss)'
--search 'astar(cpdbs(patterns=hillclimbing(max_time=900),transform=multiply_out_conditional_effects),
    symmetries=sym,pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01),num_por_probes=1000)'
```

16.
```
--symmetries 'sym=structural_symmetries(search_symmetries=oss)'
--search 'astar(zopdbs(patterns=genetic(pdb_max_size=50000,num_collections=5,num_episodes=30,
    mutation_probability=0.01), transform=multiply_out_conditional_effects),symmetries=sym,
    pruning=stubborn_sets_simple(minimum_pruning_ratio=0.01), num_por_probes=1000)'
```

Domains of the Training Set

The following lists contain all benchmark domains we used for training, named as in the repository under `https://bitbucket.org/SilvanS/ipc2018-benchmarks`. Domains with the prefix ss are either additional domains not contained in the original repository under `https://bitbucket.org/aibasel/downward-benchmarks` or copies of already present domains containing additional tasks that we generated.

STRIPS domains:

```
['airport', 'barman-opt11-strips', 'barman-opt14-strips', 'blocks',
'childsnack-opt14-strips', 'depot', 'driverlog', 'elevators-opt08-strips',
'elevators-opt11-strips', 'floortile-opt11-strips',
'floortile-opt14-strips', 'freecell', 'ged-opt14-strips', 'grid',
'gripper', 'hiking-opt14-strips', 'logistics00', 'logistics98', 'miconic',
'movie', 'mprime', 'mystery', 'nomystery-opt11-strips',
'openstacks-opt08-strips', 'openstacks-opt11-strips',
'openstacks-opt14-strips', 'openstacks-strips', 'parcprinter-08-strips',
'parcprinter-opt11-strips', 'parking-opt11-strips', 'parking-opt14-strips',
'pathways-noneg', 'pegsol-08-strips', 'pegsol-opt11-strips',
'pipesworld-notankage', 'pipesworld-tankage', 'psr-small', 'rovers',
'satellite', 'scanalyzer-08-strips', 'scanalyzer-opt11-strips',
'sokoban-opt08-strips', 'sokoban-opt11-strips', 'storage',
'tetris-opt14-strips', 'tidybot-opt11-strips', 'tidybot-opt14-strips',
'tpp', 'transport-opt08-strips', 'transport-opt11-strips',
'transport-opt14-strips', 'trucks-strips', 'visitall-opt11-strips',
'visitall-opt14-strips', 'woodworking-opt08-strips',
'woodworking-opt11-strips', 'zenotravel', 'ss_barman', 'ss_ferry',
'ss_goldminer', 'ss_grid', 'ss_hanoi', 'ss_hiking', 'ss_npuzzle',
'ss_spanner',]
```

Domains with conditional effects:

```
['briefcaseworld', 'cavediving-14-adl', 'citycar-opt14-adl', 'fsc-blocks',
'fsc-grid-a1', 'fsc-grid-a2', 'fsc-grid-r', 'fsc-hall', 'fsc-visualmarker',
'gedp-ds2ndp', 'miconic-simpleadl', 't0-adder', 't0-coins', 't0-comm',
't0-grid-dispose', 't0-grid-push', 't0-grid-trash', 't0-sortnet',
't0-sortnet-alt', 't0-uts', 'ss_briefcaseworld', 'ss_cavediving',
'ss_citycar', 'ss_maintenance', 'ss_maintenance_large', 'ss_schedule',]
```

Domain-wise Trainingset Performance

	Portfolio Components																	Delfi		Oracle
	DKS								OSS									1	2	
	blind	lmc	P1	P2	M1	M2	M3	M4	blind	lmc	P1	P2	M1	M2	M3	M4	Sym			
airport (50)	27	29	31	28	27	2	27	27	27	29	31	28	27	2	27	27	27	27	29	**32**
barman-opt11-strips (20)	8	8	8	8	8	**12**	8	8	8	8	8	8	8	**12**	8	8	10	10	8	**12**
barman-opt14-strips (14)	3	3	3	3	3	**6**	3	3	3	3	3	3	3	**6**	3	3	**6**	**6**	3	**6**
blocks (35)	21	28	28	25	28	26	26	28	21	28	28	25	28	26	26	28	**32**	**32**	**32**	**32**
briefcaseworld (50)	8	**9**	8	8	**9**	8	8	**9**	8	**9**	8	8	8	8	8	8	8	**9**	8	**9**
cavediving-14-adl (20)	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
childsnack-opt14-strips (20)	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	4	**6**	**6**	**6**
citycar-opt14-adl (20)	**18**	**18**	**18**	**18**	**18**	10	**18**	**18**	**18**	**18**	**18**	**18**	17	10	17	**18**	**18**	**18**	**18**	**18**
depot (22)	6	9	**12**	8	9	**12**	11	10	6	9	**12**	8	9	**12**	9	10	7	**12**	9	**12**
driverlog (20)	8	14	13	13	13	14	13	14	7	14	14	13	13	14	13	14	14	**15**	14	**15**
elevators-opt08-strips (30)	17	22	22	20	19	19	19	12	17	22	22	20	19	19	19	5	**25**	**25**	**25**	**25**
elevators-opt11-strips (20)	15	18	18	17	16	16	16	10	15	18	18	17	16	16	16	3	**19**	**19**	**19**	**19**
floortile-opt11-strips (20)	8	**14**	8	8	9	10	12	8	8	**14**	8	8	9	10	10	8	**14**	12	**14**	**14**
floortile-opt14-strips (20)	8	**20**	8	8	9	11	14	8	8	**20**	8	8	9	11	11	8	**20**	17	**20**	**20**
freecell (80)	20	15	21	20	21	22	22	20	20	15	21	20	21	22	22	20	25	**27**	21	**27**
fsc-blocks (14)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
fsc-grid-a1 (16)	2	2	0	1	2	2	2	2	2	2	0	1	2	2	2	2	**3**	**3**	2	**3**
fsc-grid-a2 (2)	**1**	**1**	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
fsc-grid-r (16)	**15**	**15**	0	0	**15**	**15**	**15**	**15**	**15**	**15**	0	0	**15**	**15**	**15**	**15**	1	1	6	**15**
fsc-hall (2)	**1**	**1**	0	0	**1**	**1**	**1**	**1**	**1**	**1**	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
fsc-visualmarker (7)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ged-opt14-strips (20)	15	15	19	19	19	19	19	5	15	15	19	18	18	15	15	5	19	19	**20**	19
gedp-ds2ndp (24)	18	18	4	4	**22**	18	**22**	**22**	18	18	4	4	18	14	18	18	18	**22**	16	**22**
grid (5)	1	2	**3**	2	2	**3**	2	2	1	2	**3**	2	2	**3**	**3**	2	2	**3**	2	**3**
gripper (20)	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
hiking-opt14-strips (20)	17	13	19	19	19	19	19	18	17	13	19	19	19	19	19	17	19	18	19	**20**
logistics00 (28)	12	20	**21**	**21**	20	20	20	19	12	20	20	20	20	20	20	19	19	**21**	**21**	**21**
logistics98 (35)	2	6	5	5	5	5	5	5	2	**7**	5	6	5	5	5	5	6	**7**	6	**7**
miconic (150)	56	142	61	61	84	78	61	57	56	142	61	61	85	78	61	56	109	143	142	**144**
miconic-simpleadl (150)	80	144	81	82	87	87	86	80	80	144	80	81	87	87	86	80	149	149	144	**150**
movie (30)	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**
mprime (35)	18	23	24	23	23	21	23	23	20	22	25	23	24	21	23	24	24	25	24	**26**
mystery (30)	15	18	17	17	17	16	16	17	15	18	18	17	17	16	17	17	15	18	15	**19**
nomystery-opt11-strips (20)	9	16	**20**	**20**	**20**	**20**	**20**	14	9	16	**20**	**20**	**20**	**20**	**20**	14	16	18	16	**20**
openstacks-opt08-strips (30)	24	23	24	24	24	24	23	9	24	23	24	24	24	24	23	9	**30**	**30**	**30**	**30**
openstacks-opt11-strips (20)	18	18	18	18	18	18	18	4	18	18	18	18	18	18	18	4	**20**	**20**	**20**	**20**
openstacks-opt14-strips (20)	5	3	5	5	5	5	3	0	5	3	5	5	5	5	3	0	**20**	**20**	**20**	**20**
openstacks-strips (30)	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**20**	19	7	**20**
parcprinter-08-strips (30)	**30**	**30**	29	**30**	27	9	26	**30**	**30**	**30**	29	**30**	22	9	24	**30**	22	28	**30**	**30**
parcprinter-opt11-strips (20)	**20**	**20**	**20**	**20**	**20**	5	19	**20**	**20**	**20**	**20**	**20**	17	5	17	**20**	17	**20**	**20**	**20**
parking-opt11-strips (20)	0	3	**8**	1	2	1	1	**8**	1	3	**8**	1	2	4	1	**8**	1	**8**	**8**	**8**
parking-opt14-strips (20)	0	4	7	0	5	4	1	7	0	3	**8**	0	4	4	3	**8**	3	**8**	**8**	**8**
pathways-noneg (30)	4	**5**	**5**	**5**	**5**	**5**	**5**	**5**	4	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
pegsol-08-strips (30)	28	29	**30**	28	**30**	29	29	28	28	29	**30**	29	**30**	29	29	20	29	**30**	**30**	**30**
pegsol-opt11-strips (20)	18	19	**20**	18	**20**	19	19	18	18	19	**20**	19	**20**	19	19	10	19	**20**	**20**	**20**
pipesworld-notankage (50)	20	21	**25**	23	21	4	20	20	20	21	**25**	24	21	4	20	20	15	**25**	22	**25**
pipesworld-tankage (50)	17	17	20	17	17	16	17	21	17	16	21	17	17	17	19	20	16	21	17	**22**
psr-small (50)	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**	**50**
rovers (40)	7	12	11	10	9	11	11	9	7	12	11	10	9	11	10	9	**14**	**14**	13	**14**
satellite (36)	7	**14**	9	9	9	9	9	9	7	**14**	9	9	9	9	9	9	10	**14**	11	**14**

scanalyzer-08-strips (30)	15	17	17	15	16	18	17	6	15	17	18	16	**19**	**19**	17	6	12	**19**	15	**19**
scanalyzer-opt11-strips (20)	11	14	13	11	12	14	13	3	11	14	14	12	**15**	**15**	13	3	9	**15**	12	**15**
sokoban-opt08-strips (30)	28	**30**	**30**	28	**30**	26	**30**	28	28	**30**	**30**	28	**30**	26	28	28	28	**30**	29	**30**
sokoban-opt11-strips (20)	**20**	**20**	**20**	**20**	**20**	16	**20**	19	**20**	**20**	**20**	**20**	**20**	16	**20**	19	**20**	**20**	**20**	**20**
ss_barman (33)	10	8	12	10	11	10	13	10	10	8	14	11	11	10	11	10	**24**	23	**24**	**24**
ss_briefcaseworld (110)	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**
ss_cavediving (100)	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	29	**30**	**30**	**30**	**30**	**30**
ss_citycar (288)	15	22	16	16	12	0	11	14	15	20	15	15	12	0	11	14	24	26	28	**31**
ss_ferry (132)	78	122	95	94	89	89	90	64	80	122	95	94	89	90	89	64	108	**123**	122	122
ss_goldminer (144)	50	79	**102**	53	98	62	80	50	50	80	**102**	53	98	62	97	50	75	**102**	**102**	**102**
ss_grid (108)	52	50	74	58	58	60	58	56	52	50	74	58	58	60	72	56	**91**	89	90	**91**
ss_hanoi (30)	**13**	10	**13**	**13**	**13**	**13**	**13**	**13**	**13**	10	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**
ss_hiking (112)	74	56	85	82	79	84	78	76	74	56	86	84	81	85	81	78	90	93	92	**96**
ss_maintenance (128)	0	45	65	0	10	26	11	11	0	44	65	0	9	24	8	2	0	64	64	**67**
ss_maintenance_large (100)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ss_npuzzle (30)	6	6	**12**	**12**	6	6	6	**12**	6	6	**12**	**12**	6	6	6	**12**	9	**12**	**12**	**12**
ss_schedule (168)	63	148	130	95	158	144	135	156	64	147	131	98	152	145	119	158	0	158	**163**	**163**
ss_spanner (132)	86	89	95	86	95	95	**132**	95	82	85	92	82	92	92	89	92	**132**	**132**	**132**	**132**
storage (30e)	16	17	18	16	18	18	18	17	17	17	**19**	17	18	**19**	17	18	15	**19**	16	**19**
t0-adder (2)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
t0-coins (30)	10	15	0	9	10	10	10	10	10	15	0	9	10	10	10	10	15	15	**16**	**16**
t0-comm (25)	5	6	5	5	5	5	5	5	5	6	5	5	5	5	5	5	**15**	**15**	**15**	**15**
t0-grid-dispose (15)	0	**3**	0	0	0	0	0	0	0	**3**	0	0	0	0	0	0	1	1	**3**	**3**
t0-grid-push (5)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
t0-grid-trash (1)	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
t0-sortnet (5)	**2**	**2**	0	0	**2**	0	**2**	**2**	**2**	**2**	0	0	**2**	0	**2**	**2**	0	1	**2**	**2**
t0-sortnet-alt (6)	**4**	**4**	1	1	**4**	**4**	**4**	**4**	**4**	**4**	1	1	**4**	**4**	**4**	**4**	2	2	**4**	**4**
t0-uts (29)	6	8	10	7	10	10	10	10	6	9	**11**	7	**11**	**11**	**11**	**11**	6	9	9	**11**
tetris-opt14-strips (17)	12	11	**13**	12	**13**	1	**13**	**13**	12	11	12	12	**13**	1	**13**	**13**	10	**13**	10	**13**
tidybot-opt11-strips (20)	10	**17**	15	13	11	1	11	10	10	**17**	15	13	11	1	10	10	14	**17**	**17**	**17**
tidybot-opt14-strips (20)	1	**13**	11	8	3	0	3	2	1	**13**	11	8	3	0	2	2	6	**13**	12	**13**
tpp (30)	7	8	7	7	9	**12**	8	7	7	8	7	8	9	11	8	7	8	11	11	**12**
transport-opt08-strips (30)	11	11	11	12	11	11	11	11	11	11	12	12	11	11	12	11	**14**	**14**	13	**14**
transport-opt11-strips (20)	6	7	7	8	6	7	6	6	6	7	8	8	7	7	8	7	**10**	9	**10**	**10**
transport-opt14-strips (20)	7	6	7	7	7	7	7	7	7	6	7	7	7	7	7	7	**9**	**9**	**9**	**9**
trucks-strips (30)	9	**12**	11	10	9	10	10	9	9	**12**	11	10	9	10	10	9	**12**	**12**	11	**12**
visitall-opt11-strips (20)	9	12	16	13	9	10	9	16	9	12	14	12	9	10	9	14	12	**17**	**17**	**17**
visitall-opt14-strips (20)	3	6	12	7	4	4	4	12	3	6	8	6	4	4	4	8	7	13	**14**	**14**
woodworking-opt08-strips (30)	22	**30**	23	22	**30**	**30**	**30**	28	22	**30**	23	22	**30**	**30**	22	28	28	**30**	**30**	**30**
woodworking-opt11-strips (20)	16	**20**	17	16	**20**	**20**	**20**	**20**	16	**20**	17	16	**20**	**20**	16	**20**	**20**	**20**	**20**	**20**
zenotravel (20)	8	**13**	12	11	12	12	11	11	8	**13**	12	11	12	**13**	11	11	11	12	12	**13**
Sum (3721)	1470	1956	1836	1602	1796	1645	1767	1615	1472	1948	1838	1606	1782	1643	1707	1568	1867	2282	2236	**2350**

Table 4: Coverage of the trainingset. Abbreviations: lmc: LM-cut; P1: HC-PDB; P2: GA-ZOPDB; M1: B-SCCdfp; M2: B-MIASMdfp; M3: B-sbMIASM; M4: G-SCCdfp; Sym: SymBA* 2014; Orcl: oracle portfolio over all component planners.