Automatic Configuration of Sequential Planning Portfolios

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

Abstract

Sequential portfolios have been demonstrated to be very powerful at exploting the complementary strength of different planners. Similarly, automated algorithm configuration can find high-performance customized instantiations of a parameterized planning framework. Although some work has been done towards combining algorithm configuration and portfolios, the problem of automatically generating a sequential portfolio from a parameterized algorithm for a given type of planning tasks is unsolved. Here, we present an approach for this problem that greedily generates the (planner configuration, runtime) pair that, when appended to the current portfolio, maximizes portfolio improvement per runtime spent. This method yields portfolios that are provably within a constant factor of optimal and yields promising results compared to previous portfolios from the literature.

Introduction

Over the years the automated planning community has created a very large number of different planning algorithms. However, none of them dominates all others on all planning tasks. Since automatically choosing the best planner for a given task remains a mostly unsolved problem, today's most successful planners run a *sequential portfolio* of individual planners (Coles et al. 2012). A prototypical example of a planner portfolio is Fast Downward Stone Soup (FDSS) (Helmert, Röger, and Karpas 2011), which won the deterministic optimizing track of the 2011 International Planning Competition (IPC) and was the runner-up in the deterministic satisficing track.

Recently, several approaches have been developed for automatically constructing portfolios. Given a finite set of planners $\mathcal{A}$ and their performance on a benchmark set Π, the aspeed system (Hoos et al. 2012a) constructs the optimal sequential portfolio of algorithms in $\mathcal{A}$ for Π; however, this is an $\mathcal{NP}$-hard problem and aspeed's runtime requirement can be very large. Streeter and Smith (2008) introduced an efficient greedy algorithm that constructs a portfolio guaranteed to be within a constant factor of the optimal one. For a more general overview of the work on sequential portfolios in the automated planning community we refer to Vallati (2012).

While planning portfolios combine the complementary strengths of existing planners on a heterogeneous set of benchmarks, performance on a particular homogeneous type of instances can often be improved by tuning a planner's parameters. Specifically, applied to highly parameterized planning systems, automated *algorithm configuration* has recently been shown to find novel planner instantiations that are customized to yield the highest available performance on particular types of benchmark instances (Fawcett et al. 2011; Vallati et al. 2013).

Due to their respective success, several recent lines of work found in the Satisfiability Testing (SAT) literature use algorithm configuration to construct portfolios of a single parameterized algorithm based on algorithm selection (Rice 1976) and on parallel execution of solvers (Huberman, Lukose, and Hogg 1997). The portfolio construction procedure Hydra (Xu, Hoos, and Leyton-Brown 2010) uses algorithm configuration to determine configurations of a highly parameterized algorithm to be used as component solvers in the SATzilla portfolio-based *algorithm selector* (Xu et al. 2008). Likewise, the GREEDY method in Hoos et al. (2012b) uses algorithm configuration to determine parameter configurations for a *parallel portfolio*. However, there does not yet exist a general procedure for constructing sequential portfolios from a highly parameterized algorithm.

In this paper, we present such a method with the objective of performing well for a given heterogeneous set of benchmarks Π. A special case of this problem has already been considered by Seipp et al. (2012a), who considered a set Π comprised of 21 known subsets, $\Pi = \Pi_1 \cup \cdots \cup \Pi_{21}$. They first used algorithm configuration to identify a customized configuration $\mathcal{C}_i$ of a planning system for each known subset Π_i and in a second step used several different methods for combining $\{\mathcal{C}_1, \ldots, \mathcal{C}_{21}\}$ in a sequential portfolio. Their results suggest that given enough time, it is best to run $\{\mathcal{C}_1, \ldots, \mathcal{C}_{21}\}$ with equal time shares; however, for shorter runtimes better results were obtained by only running a subset of them, and by running some configurations longer than others (Seipp et al. 2012b).

Our work removes the need to know which subsets the benchmark set Π is comprised of, and the need for a two-step process: rather than optimizing separately for known homogeneous subsets $\Pi_1, \ldots, \Pi_{21}$ of Π, we directly optimize portfolio performance for the entirety of Π. Specifically, starting with an empty portfolio, we use algorithm configuration to greedily find (configuration, time slice) pairs that, when appended to the current portfolio, produce the high-

Algorithm 1 Configure a portfolio for instances Π, space of algorithms $\mathcal{A}$ and total portfolio runtime T, using a budget B for each call to OPTIMIZE.

```
function ConfigurePortfolio($\Pi$, $\mathcal{A}$, $T$)
    $P \leftarrow \langle\rangle$
    $T_{\text{used}} \leftarrow 0$
    while $r = T - T_{\text{used}} > 0$ do
        $\langle a, t\rangle \leftarrow$ Optimize($P$, $\mathcal{A}$, $\Pi$, $r$, $B$)
        $\Pi \leftarrow \Pi \setminus \{\pi \in \Pi \mid q(\langle a, t\rangle, \pi) = 1\}$
        if $\Pi$ reached fixpoint then return $P$
        $P \leftarrow P \oplus \langle a, t\rangle$
        $T_{\text{used}} \leftarrow T_{\text{used}} + t$
    return $P$
```

est gain (number of additionally solved tasks for optimal planning; quality improvement for satisficing planning) per time unit spent. Invoking theoretical results by Streeter and Smith (2008), we can prove that this conceptually simple procedure yields sequential portfolios whose performance is within a constant factor of the optimal portfolio, concerning both the expected number of benchmarks solved in any given time T and the expected runtime for solving a benchmark. Experiments show that our procedure also constructs strong portfolios in practice. Specifically, for satisficing planning we achieve better performance on the training benchmark set Π than any previous portfolio.

Definitions

Informally, a *classical planning task* π consists of an initial state, a set of goal states and a set of operators. Solving π implies finding an operator sequence that leads from the initial state to a goal state. While *optimal planning* is the task of finding the solution with the minimum sum of operator costs, in *satisficing planning* any solution is accepted. Let $\mathcal{A}$ be a possibly infinite set of *planning algorithms*. Then we define $c(\langle a, t\rangle, \pi)$ as the *cost* of the solution $a \in \mathcal{A}$ finds on planning task π within time t, or ∞ if it does not find a solution. Furthermore we let $c^\star(\pi)$ be the *minimum solution cost* for task π. Following the evaluation criteria from the IPC for satisficing planners we define the *solution quality* $q(\langle a, t\rangle, \pi) = \frac{c^\star(\pi)}{c(\langle a,t\rangle,\pi)}$ as the minimum known solution cost divided by the solution cost achieved by a in time t. Note that in optimal planning this quality is either 0 or 1. The pair $\langle a, t\rangle$ is said to *solve* a task π if $q(\langle a, t\rangle, \pi) = 1$.

Next we need to define the notion of sequential portfolios.

Definition 1. Sequential portfolios.
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 a is allowed to run for.*

The *quality* $q(P, \pi)$ *of a portfolio* on planning task π is the maximum quality any component $\langle a, t\rangle$ of P achieves on π. A portfolio P solves a task π if any of its components $\langle a, t\rangle$ solves it. We denote the portfolio resulting from appending a component $\langle a, t\rangle$ to a portfolio P by $P \oplus \langle a, t\rangle$.

Configuration of Sequential Portfolios

We outline our algorithm that automatically configures portfolios for satisficing and optimal planning in Alg. 1. It takes as input a set of training instances Π, a configuration space of planning algorithms $\mathcal{A}$, and the time T that the portfolio is allowed to run. The procedure starts from an empty portfolio P and iteratively extends it. After appending a (planner, time slice) pair $\langle a, t\rangle$ to P, we remove all tasks π from Π that were solved by $\langle a, t\rangle$ in order to focus on other, unsolved instances in the next iteration. The procedure terminates when the sum of P 's components' time slices reaches the available time T or when the chosen best pair $\langle a, t\rangle$ does not solve any additional tasks. Note that such a pair $\langle a, t\rangle$ will not be added to the portfolio.

We select the next (planner, time slice) pair $\langle a, t\rangle$ with the subsidiary procedure OPTIMIZE. Intuitively, OPTIMIZE aims to maximize the marginal improvement of portfolio performance per time spent. More formally, it tries to maximize the performance metric

$$m_P(\langle a, t\rangle, \Pi) = \frac{q(P \oplus \langle a, t\rangle, \Pi) - q(P, \Pi)}{t}. \tag{1}$$

Note that for optimal planning this means we seek to maximize the number of newly solved tasks per time spent, whereas for satisficing planning it means a maximal increase in quality per time spent.

While Alg. 1 also works for a finite set of pre-specified planner configurations, as noted above, $\mathcal{A}$ is the configuration space of a parameterized planning framework in our case. As the number of candidate algorithms and time limits of such a configuration space is infinite, we use an algorithm configuration procedure to automatically find pairs $\langle a, t\rangle$ that maximize the performance metric in Eq. 1.

Since algorithm configuration procedures evaluate the performance of candidate configurations on single instances at a time, in order to optimize the performance metric in Eq. 1, we must define a performance metric $m'_P(\langle a, t\rangle, \pi)$ such that $m_P(\langle a, t\rangle, \Pi) = \sum_{\pi\in\Pi} m'_P(\langle a, t\rangle, \pi)$. Fortunately, this is trivial for the additive qualities we aim to optimize in satisficing and optimal planning by setting $m'_P(\langle a, t\rangle, \pi) = m_P(\langle a, t\rangle, \{\pi\})$. Similar to what was done in the Hydra portfolio construction procedure (Xu, Hoos, and Leyton-Brown 2010), we can cache the computation of this marginal improvement $m'_P(\langle a, t\rangle, \pi)$ of a pair $\langle a, t\rangle$ over a portfolio P on an instance π by remembering the best quality $q_{\text{old}}(\pi)$ the current portfolio P achieves on each instance $\pi \in \Pi$; to evaluate $m'_P(\langle a, t\rangle, \pi)$, we must then only run a on π for time t and compare the result with $q_{\text{old}}(\pi)$. Note that this means the cost of a single evaluation of $\langle a, t\rangle$ is bound by t. This is a key difference to the Hydra construction mechanism (Xu, Hoos, and Leyton-Brown 2010) or the greedy construction of parallel portfolios by Hoos et al. (2012b), which have to run a with the total time budget T allocated for the portfolio. Since t is typically only a small fraction of T, our subsidiary algorithm configuration procedure can make much more progress in the same amount of time than when being used by these other portfolio construction procedures.

Theoretical Analysis

We now analyze our algorithm theoretically, using results by Streeter and Smith (2008), who studied a special case of Alg. 1 where the sets $\mathcal{A}$ and Π are finite and we know the time required by every planner $a \in \mathcal{A}$ on every planning task $\pi \in \Pi$. Defining the expected runtime of a portfolio on benchmark set Π as its average time to success, Streeter and Smith (2008) showed that

- for any $\epsilon > 0$, it is $\mathcal{NP}$-hard to find a portfolio guaranteed to have an expected runtime bounded by a factor of $4 - \epsilon$ times the expected runtime of the optimal portfolio;
- it is $\mathcal{NP}$-hard to find a portfolio whose quality within a time budget T is guaranteed to be bounded by a factor of $1 - 1/e$ times the quality of the optimal portfolio with time budget T; and
- the greedy algorithm finds portfolios with approximation ratios to the optimal portfolio that are simultaneously tight for both expected runtime and quality.

Given a *perfect* algorithm configuration procedure, their results (Theorems 2 and 3 of Streeter and Smith (2008)) directly apply to prove the following theoretical guarantees for Alg. 1:

Theorem 1. *Let q^* be the highest possible quality a portfolio comprised of configurations from $\mathcal{A}$ can achieve in time T on benchmark set Π, and let r^* be the lowest expected runtime on Π a portfolio comprised of configurations from $\mathcal{A}$ can achieve. Given a procedure* OPTIMIZE *that returns a maximizer $\langle a, t\rangle$ of Eq. 1, Alg. 1 will then construct a portfolio with quality bounded by $(1 - (1/e)) \cdot q^*$ and expected runtime bounded by $4 \cdot r^*$.*

We note that this theorem only provides a guarantee for the set (or distribution) of benchmarks Π used for training the portfolio. When the learned portfolio is used for another benchmark set Π', all depends on how similar Π and Π' are. If the tasks in Π', for example, come from domains with very different characteristics than those in Π or are consistently harder, then the portfolio may not perform well. We note that this problem of generalization is not unique to automatic portfolio design; a planner (or even portfolio) manually designed for high performance on Π would have exactly the same problem. We also note that in practice, OPTIMIZE will only yield the truly optimal portfolio component $\langle t, a\rangle$ rarely, especially when searching over combinatorial parameter spaces with 50+ categorical parameters. However, as we will show in the experiments, it often finds portfolio components that yield strong portfolio performance.

Experiments

We now study our automated portfolio configuration algorithm empirically. In order to directly compare against the most closely related work by Seipp et al. (2012a), we focus on satisficing planning (the only kind considered by them) and use the same setup (including the same parameterized planning framework and the same planning tasks).

The parameterized framework for satisficing planning Seipp et al. (2012a) used is that of the Fast Downward (FD) planning system (Helmert 2006). We refer to Fawcett et al. (2011) for a detailed description of its numerous search algorithms and heuristics, but note briefly that this framework comprises 70 parameters, which, when discretized, give rise to 1.935×10^{26} possible configurations.

The benchmark set Π used by Seipp et al. (2012a) comprises a total of 724 tasks from 21 domains from the IPC 1998–2006 challenges. Seipp et al. (2012a) exploit knowledge about this composition by configuring custom planners for each of the 21 domains. In contrast, our automated portfolio construction mechanism does not require this knowledge and operates directly on Π. As Seipp et al. (2012a), we follow the IPC rules by setting the total portfolio limit to 30 minutes and aborting each component planner if it uses more than 2 GB of memory.

As the OPTIMIZE procedure of our algorithm, we use the algorithm configuration procedure SMAC (Hutter, Hoos, and Leyton-Brown 2011) because of its ability to handle both categorical and numerical parameters of very large configuration spaces. To evaluate the impact of better configuration procedures, we evaluate our algorithm with different time budgets B for SMAC.

Table 1 gives an overview over the training performance of two sets of portfolios learned for satisficing planning: one optimizes coverage (ie uses qualities of 0 or 1 in OPTIMIZE), the other one optimizes quality as described in the previous section. Both variants were tested with different budgets $B \in \{1\text{h}, 2\text{h}, 5\text{h}, 10\text{h}\}$. We compare against the performance of satisficing FDSS (1 and 2) and the portfolio generators presented in Seipp et al. (2012a): Cluster (cluster), Domain-wise (dw), Increasing Time Limit (inc), Random Iterative Search (ris), Selector (sel), Stone Soup (ss) and the uniform portfolio (uniform). Analyzing the last two blocks of the table with our portfolios, we can observe that the training performance of our portfolios increases as expected with the time allowed for each configuration run. Compared against each other, there is a clear advantage of optimizing for quality and not for coverage. When comparing against the other approaches, we observe that our portfolios optimizing quality with budgets of 5h and 10h achieve a better performance than any other portfolio.

Figure 1 shows how quality improves during the portfolio configuration. Each point represents the time at which the portfolio switches to a new configuration and the corresponding achieved quality at that time. The displayed qualities represent those learned at configuration time.[1] We observe that higher time budgets for the configuration procedure result in higher portfolio performance, especially in the early stages of the portfolio. Not surprisingly, for all budgets, quality improves quickly in the beginning and then slows down as it becomes harder and harder to make marginal improvements over the portfolio so far.

[1] In contrast, Table 1 evaluates the portfolios as anytime-search algorithms, where each iteration except the first uses the cost of the solution from the previous iteration as a maximum bound on g-values during search, thus possibly improving the overall quality. This explains why our 5h and 10h portfolios achieve the same quality after the configuration process, but have a distinct evaluation quality.

	FDSS		Seipp et al. (2012a)							Objective: Coverage				Objective: Quality			
Quality	**1**	**2**	**cluster**	**dw**	**inc**	**ris**	**sel**	**ss**	**uniform**	**1h**	**2h**	**5h**	**10h**	**1h**	**2h**	**5h**	**10h**
airport (39)	31.82	31.69	32.69	30.71	31.52	30.85	31.71	32.85	32.69	28.69	31.82	28.66	30.96	29.76	31.89	31.91	**32.89**
depot (20)	15.26	14.52	17.46	16.79	15.85	17.32	17.32	17.34	**18.41**	15.61	17.06	17.63	16.56	16.19	17.07	17.02	17.46
driverlog (9)	8.33	7.83	8.41	7.93	7.82	8.47	8.41	8.43	8.28	8.25	7.98	8.13	8.10	8.09	8.32	8.36	**8.50**
freecell (78)	71.45	69.35	74.68	72.13	71.92	74.67	74.55	74.55	74.78	73.20	74.59	74.91	74.84	74.58	75.00	75.33	**76.28**
grid (5)	4.56	3.91	4.74	4.64	4.51	4.74	4.74	4.74	4.74	4.70	4.93	4.93	4.95	4.98	**5.00**	4.97	4.97
logistics-00 (28)	27.43	26.63	27.77	27.76	27.63	27.90	27.84	27.83	27.80	27.61	27.59	27.84	27.89	27.91	**27.97**	27.90	27.94
miconic-full (90)	75.72	75.86	76.09	75.99	67.87	76.14	76.21	76.09	75.93	71.83	72.43	73.03	70.79	77.69	**78.66**	78.23	78.50
mprime (26)	**26.00**	25.07	**26.00**	**26.00**	**26.00**	**26.00**	25.92	25.92	**26.00**	25.69	**26.00**	**26.00**	**26.00**	**26.00**	**26.00**	**26.00**	**26.00**
opt-telegraphs (47)	17.00	5.00	17.00	8.00	12.00	14.00	20.00	14.00	16.00	4.00	11.00	11.00	19.00	20.00	22.00	**47.00**	**47.00**
pathways (17)	16.71	16.04	16.75	16.70	16.81	16.81	16.73	16.71	16.73	16.74	16.69	16.77	16.58	16.85	**16.86**	16.84	16.84
philosophers (43)	43.00	43.00	43.00	43.00	43.00	43.00	43.00	43.00	43.00	43.00	43.00	43.00	43.00	43.00	43.00	43.00	43.00
pipesworld-nt (43)	33.28	34.21	36.39	34.02	33.14	35.78	**36.61**	35.41	35.37	35.45	36.02	36.04	36.45	35.93	36.16	36.15	36.21
pipesworld-t (47)	33.96	33.17	38.11	37.11	35.33	**38.47**	37.26	**38.47**	37.92	34.82	35.43	34.90	37.20	33.96	34.39	34.90	37.63
psr-large (42)	21.87	22.03	24.12	**25.09**	23.09	23.06	25.03	23.00	24.06	20.24	22.45	22.56	22.77	19.89	21.43	21.48	22.45
rovers (21)	19.83	18.56	20.23	20.11	20.10	20.26	20.21	20.25	20.23	19.29	20.14	19.60	19.35	20.22	20.15	20.28	**20.48**
satellite (22)	21.76	21.31	21.66	21.61	21.57	21.68	21.67	21.67	21.63	20.67	20.72	20.54	21.26	19.92	21.42	20.93	**21.91**
schedule (80)	74.02	69.75	76.49	76.23	76.16	76.60	75.96	76.36	76.51	74.09	73.16	74.00	74.26	76.66	77.27	**79.60**	78.61
storage (20)	11.86	10.64	9.90	9.60	9.60	9.90	9.90	9.90	9.90	10.72	9.70	10.72	11.61	10.44	9.77	10.48	**11.87**
tpp (17)	15.66	13.87	15.83	15.21	15.55	15.64	15.82	15.83	15.87	15.22	15.21	15.25	15.11	15.80	**16.00**	15.79	15.98
trucks (23)	12.12	11.38	15.17	13.09	10.89	17.41	15.30	15.36	15.11	15.48	15.54	15.27	**18.78**	14.58	13.41	16.38	15.25
zenotravel (7)	6.54	6.47	6.73	6.63	6.25	6.73	6.73	6.73	6.71	6.52	6.39	6.57	6.84	6.87	6.87	**6.89**	6.85
Sum (724)	588.21	560.29	609.23	588.34	576.60	605.41	610.92	604.43	607.66	571.81	587.86	587.35	602.29	599.32	608.66	639.46	**646.62**

Table 1: Quality of our portfolios compared to FDSS and portfolios from Seipp et al. (2012a) for the IPC 1998–2006 domains.

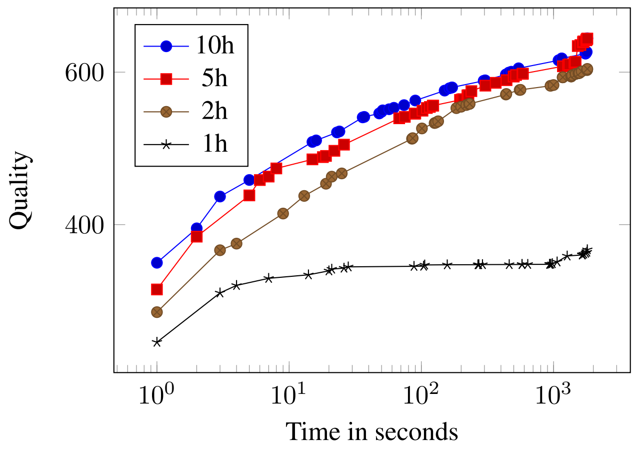

Figure 1: Total quality of the portfolios optimizing additional quality per time as a function of the portfolio time going from 0 to 18000 seconds.

Although this paper focuses on automated construction portfolio for a given benchmark set Π, one may wonder how well the portfolios generalize beyond Π. Seipp et al. (2012a) evaluated generalization performance on the IPC 2011 benchmark set. These benchmarks are quite different since no domain in Π contains action costs (whereas some domains in IPC 2011 do) and generally, problems from the IPC benchmark sets have become substantially more difficult over the years. Consequently, Seipp et al. (2012a) demonstrated that the portfolio with the best training performance on benchmark set Π is *not* the best for the IPC 2011 data, and the same holds in our case. Although it had the highest performance on Π, our 10h quality optimized portfolio achieved a test quality of 231.90, which is better than FDSS 1 (207.52) and FDSS 2 (193.52), but worse than the uniform portfolio (247.93) of Seipp et al. (2012a).

We also evaluated our method for optimal planning, using 21 benchmark domains amounting to 835 solvable tasks from the IPC 1998–2006 challenges. In this case, our automated method only solved 476 instances, compared to the 491 instances solved by FDSS, the winner of the optimal planning track in the IPC 2011. We believe this is because our OPTIMIZE procedure SMAC did not have enough time to find good enough configurations. Due to the higher runtimes needed to solve instances in optimal planning, SMAC only managed to execute roughly one third of the FD runs it executed for satisficing planning. Combined with this hurdle, it is not surprising that 10h of automated algorithm configuration is too short to find configurations as good as those manually defined by the leading human experts who constructed FDSS.

Conclusion

In this work, we presented a novel method for combining the strengths of sequential portfolios for planning and the possibility of using algorithm configuration to automatically find good configurations from a given parameter space. By incorporating the timeout for each portfolio component into the configuration space and optimizing for the highest portfolio improvement per runtime spent, we further reduced the required CPU time for the overall configuration process. Experiments showed promising results, especially for satisficing planning.

In future work, we plan to study the case of optimal planning in more detail. We also plan to use our algorithm to configure portfolios on the combined configuration spaces of multiple different planning systems. To facilitate generalization to unknown test sets, we plan to configure portfolios on a much broader class of planning tasks. We also plan to configure portfolios for the setup of the IPC learning track, where test and training sets can be expected to be very similar.

References

Coles, A.; Coles, A.; García Olaya, A.; Jiménez, S.; Linares López, C.; Sanner, S.; and Yoon, S. 2012. A survey of the Seventh International Planning Competition. *AI Magazine* 33(1):83–88.

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.; Röger, G.; and Karpas, E. 2011. Fast Downward Stone Soup: A baseline for building planner portfolios. In *ICAPS 2011 Workshop on Planning and Learning*, 28–35.

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

Hoos, H.; Kaminski, R.; Schaub, T.; and Schneider, M. T. 2012a. aspeed: ASP-based solver scheduling. In Dovier, A., and Costa, V. S., eds., *ICLP (Technical Communications)*, volume 17 of *LIPIcs*, 176–187. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Hoos, H.; Leyton-Brown, K.; Schaub, T.; and Schneider, M. 2012b. Algorithm configuration for portfolio-based parallel SAT-solving. In Coletta, R.; Guns, T.; O'Sullivan, B.; Passerini, A.; and Tack, G., eds., *Proceedings of the 1st Workshop on COmbining COnstraint solving with MIning and LEarning (CoCoMile 2012)*, 7–12.

Huberman, B. A.; Lukose, R. M.; and Hogg, T. 1997. An economics approach to hard computational problems. *Science* 265:51–54.

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.

Rice, J. R. 1976. The algorithm selection problem. *Advances in Computers* 15:65–118.

Seipp, J.; Braun, M.; Garimort, J.; and Helmert, M. 2012a. 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.; Braun, M.; Garimort, J.; and Helmert, M. 2012b. Learning portfolios of automatically tuned planners: Detailed results. Technical Report 268, Albert-Ludwigs-Universität Freiburg, Institut für Informatik.

Streeter, M. J., and Smith, S. F. 2008. New techniques for algorithm portfolio design. In *Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence (UAI 2008)*, 519–527.

Vallati, M.; Fawcett, C.; Gerevini, A.; Hoos, H. H.; and Saetti, A. 2013. Automatic generation of efficient domain-optimized planners from generic parametrized planners. In Helmert, M., and Röger, G., eds., *Proceedings of the Sixth Annual Symposium on Combinatorial Search (SoCS 2013)*, 184–192. AAAI Press.

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.

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.