Algorithm Selector and Prescheduler in the ICON challenge

François Gonard, Marc Schoenauer, Michèle Sebag

Algorithm Selector and Prescheduler in the ICON challenge

François Gonard, Marc Schoenauer, and Michèle Sebag

Acknowledgements This work has been carried out in the framework of IRT SystemX and therefore granted with public funds within the scope of the French Program "Investissements d'Avenir".

Abstract Algorithm portfolios are known to offer robust performances, efficiently overcoming the weakness of every single algorithm on some particular problem instances. Two complementary approaches to get the best out of an algorithm portfolio is to achieve algorithm selection (AS), and to define a scheduler, sequentially launching a few algorithms on a limited computational budget each. The presented *Algorithm Selector And Prescheduler* system relies on the joint optimization of a pre-scheduler and a *per instance* AS, selecting an algorithm well-suited to the problem instance at hand. ASAP has been thoroughly evaluated against the state-of-the-art during the ICON challenge for algorithm selection, receiving an honourable mention. Its evaluation on several combinatorial optimization benchmarks exposes surprisingly good results of the simple heuristics used; some extensions thereof are presented and discussed in the paper.

Key words: Algorithm selection, Algorithm portfolios, Combinatorial optimization

François Gonard
IRT SystemX, 8 avenue de la Vauve F-91127 Palaiseau Cedex

François Gonard · Marc Schoenauer · Michèle Sebag
LRI, CNRS, INRIA, Univ. Paris-Sud, Univ. Paris-Saclay, Bat. 660 Claude Shannon F-91405 Orsay Cedex, e-mail: firstname.lastname@inria.fr

1 Introduction

In quite a few domains related to combinatorial optimization, such as satisfiability, constraint solving or operations research, it has been acknowledged for some decades that there exists no universal algorithm, dominating all other algorithms on all problem instances [21]. This result has prompted the scientific community to design algorithm portfolios addressing the various types of difficulties involved in the problem instances, *i.e.*, such that at least one algorithm in the portfolio can efficiently handle any problem instance [8, 6]. Algorithm portfolios thus raise a new issue, that of selecting *a priori* an algorithm well suited to the application domain [12]. This issue, referred to as *Algorithm Selection* (AS) [19], is key to the successful transfer of algorithms outside of research labs. It has been tackled by a number of authors in the last years [16, 9, 24, 17, 15] (more in section 2).

Algorithm selection comes in different flavors, depending on whether the goal is to yield an optimal performance in expectation with respect to a given distribution of problem instances (global AS), or an optimal performance on a particular problem instance (*per instance* AS). Note that the joint problems of selecting an algorithm and the optimal hyper-parameters thereof, referred to as *Algorithm Configuration* (AC), are often considered together in the literature, as the choice of the hyper-parameter values governs the algorithm performance. Only Algorithm Selection will be considered in the following; the focus is on the *per-instance* setting, aimed at achieving peak performance on every problem instance.

Noticing that some problem instances can be solved in no time by some algorithms, it makes sense to allocate a fraction of the computational budget to a *pre-scheduler*, sequentially launching a few algorithms with a small computational budget each. The pre-scheduler is expected to solve "easy" instances in a first stage; in a second stage, AS is only launched on problem instances which have not been solved in the pre-scheduler phase. Note that the pre-scheduler yields some additional information characterizing the problem at hand, which can be used together with the initial information about the problem instance, to support the AS phase.

This paper presents the *Algorithm Selector And Prescheduler* system (ASAP), aimed at algorithm selection in the domain of combinatorial optimization (Section 3). The main contribution lies in the joint optimization of both a pre-scheduler and a per-instance algorithm selector. The extensive empirical validation of ASAP is conducted on the ICON challenge on algorithm selection [10]. This challenge leverages the Algorithm Selection library [1], aimed at the fair, comprehensive and reproducible benchmarking of AS approaches on 13 domains ranging from satisfiability to operations research (Section 4). The comparative empirical validation of ASAP demonstrates its good performances comparatively to state-of-art pre-schedulers and AS approaches (Section 5), and its complementarity with respect to the prominent zilla algorithm (based on SATzilla [23]). The paper concludes with a discussion of the limitations of the ASAP approach, and some perspectives for further research.

2 Related work

2.1 Algorithm selectors

The algorithm selection issue, aimed at selecting the algorithm best suited to the problem at hand, was first formalized by Rice [19]. Given a problem space mapping each problem instance onto a description $\mathbf{x}$ thereof (usually $\mathbf{x}$ in $\mathbf{R}^d$) and the set $\mathscr{A}$ of algorithms in the portfolio, let $\mathscr{G}(\mathbf{x},a)$ be a performance model estimating the performance of algorithm a onto problem instance $\mathbf{x}$ for each ($\mathbf{x}$, a) pair. Such a performance model yields an AS strategy, by selecting for problem instance $\mathbf{x}$ the algorithm a with optimal $\mathscr{G}(\mathbf{x},a)$.

$$AS(\mathbf{x}) = \arg\max_{a\in\mathscr{A}}\{\mathscr{G}(\mathbf{x},a)\} \tag{1}$$

The performance model is usually built by applying machine learning approaches onto a dataset reporting the algorithm performances on a comprehensive set of benchmark problem instances (with the exception of [5], using a multi-armed bandit approach). Such machine learning approaches range from k-nearest neighbors [16] to ridge regression [23], random forests [24], collaborative filtering [20, 15], or learning to rank approaches [17]. The interested reader is referred to [11] for a more comprehensive review of algorithm selectors.

As expected, the efficiency of the machine learning approaches critically depends on the quality of the training data: i.e. the representativity of the problem instances used to train the performance model and the description of the problem instances. Considerable care has been devoted to the definition of descriptive features in the SAT and Constraint domains [22].

2.2 Schedulers

An algorithm portfolio can also take advantage of parallel computer architectures by launching several algorithms working independently or in cooperation on the considered problem instance (see, e.g., [25, 9]). Schedulers embed parallel solving strategies within a sequential setting, by defining a sequence of κ (algorithm a_i, time-out τ_i) pairs, such that each problem instance is successively tackled by algorithm a_i with a computational budget τ_i, until being solved. Note that the famed restart strategy − launching a same algorithm with different random seeds or different initial conditions − can be viewed as a particular case of scheduling strategy [6]. Likewise, AS can be viewed as a particular case of scheduler with $\kappa = 1$ and τ_1 set to the overall computational budget.

A multi-stage process, where a scheduler solves easy instances in a first stage, and remaining instances are handled by the AS and tackled by the selected algorithm

in the next stage, is described by [24]. In [9], hybrid *per-instance* schedules are proposed, with AS as one of the components.

3 Overview of ASAP

After discussing the rationale for the presented approach, this Section presents the pre-scheduler and AS components forming the ASAP system, Versions 1 and 2.

3.1 Analysis

It is notorious that the hardness of a problem instance often depends on the considered algorithm. As shown on Fig. 1 in the case of the SAT11-HAND dataset (Section 4), while several algorithms might solve 20% of the problem instances within seconds, the oracle (selecting the best one out of these algorithms for each problem instance) solves about 40% of the problem instances within seconds. Along this line, the pre-scheduler problem thus consists of selecting a few algorithms, such that running each of these algorithms for a few seconds would solve a significant fraction of the problem instances.

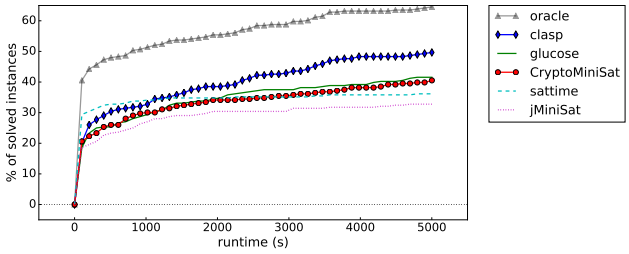

Fig. 1 Percentage of solved instances vs. runtime on the SAT11-HAND dataset, for 5 algorithms and the oracle (selecting the best algorithm out of 5 for each problem instance).

Definition 1 (Pre-scheduler). Let $\mathscr{A}$ be a set of algorithms. A κ-pre-scheduler component, defined as a sequence of κ (algorithm a_i, time-out τ_i) pairs,

$$((a_i,\tau_i)_{i=1}^{\kappa}) \text{ with } (a_i,\tau_i) \in \mathscr{A} \times \mathbb{R}^+, \ \forall i \in 1,\ldots,\kappa$$

sequentially launches algorithm a_j on any problem instance $\mathbf{x}$ until either a_j solves $\mathbf{x}$, or time τ_j is reached, or a_j stops without solving $\mathbf{x}$. If $\mathbf{x}$ has been solved, the execution stops. Otherwise, j is incremented while $j \leq \kappa$.

A pre-scheduler can contribute to better peak performances [13]. It can also increase the overall robustness of the resolution process and mitigate the impact of

AS failures (where the selected algorithm requires much computational resources to solve a problem instance or fails to solve it), as it increases the chance for each problem instance to be solved in no time, everything else being equal.

Accordingly, the ASAP system involves: i) a pre-scheduler aimed at solving as many problem instances as possible in a first stage; and ii) an AS taking care of the remaining instances. A first decision regards the division of labor between both components: how to split the available runtime between the two, and how many algorithms are involved in the pre-scheduler (parameter κ). For simplicity and tractability, the maximal runtime allocated to the pre-scheduler is fixed to T_{ps}^{max} (10% of the overall computational budget in the experiments, Section 5), and the number κ of algorithms in the pre-scheduler is set to 3. A similar setup is used in [9, 14], with the difference that the pre-scheduler uses a prescribed fraction of the computational budget.

Given T_{ps}^{max} and κ, ASAP tackles the optimization of the pre-scheduler and the AS components. Both optimization problems are interdependent: the AS must focus on the problem instances which are not solved by the pre-scheduler, while the pre-scheduler must symmetrically focus on the problem instances which are most uncertain or badly addressed by the AS. Formally, this interdependence is handled as follows:

- A performance model $\mathscr{G}(\mathbf{x}, a)$ is built for each algorithm over all training problem instances, defining AS_{init} (Eq. 1);
- A pre-scheduler is built to optimize the joint performance (pre-scheduler, AS_{init}) over all training problem instances;
- Another performance model $\mathscr{G}_2(\mathbf{x}, a)$ is built over all training problem instances, using an additional boolean feature that indicates for each problem instance whether it was solved by the above pre-scheduler; let AS_{post} denote the AS based on performance model $\mathscr{G}_2(\mathbf{x}, a)$.

ASAP finally is composed of the pre-scheduler followed by AS_{post}.

3.2 ASAP.V1 pre-scheduler

Let $(a_i, \tau_i)_{i=1}^{\kappa}$ denote a pre-scheduler, with overall computational budget $T_{ps} = \sum_{i=1}^{\kappa} \tau_i$, and let $\mathscr{F}\left((a_i, \tau_i)_{i=1}^{\kappa}\right)$ denote the associated domain-dependent performance (e.g., number of solved instances or time-to-solution). ASAP.V1 considers for simplicity equal time-outs ($a_i = \frac{1}{\kappa} T_{ps}, i = 1 \ldots \kappa$). The pre-scheduler is thus obtained by solving the following optimization problem:

$$\max_{T_{ps} \leq T_{ps}^{max}, a_1, \ldots a_\kappa} \left\{ \mathscr{F}\left((a_i, \frac{T_{ps}}{\kappa})_{i=1}^{\kappa} \right) \right\} \tag{2}$$

This mixed optimization problem is tackled in a hierarchical way, determining for each value of T_{ps} the optimal κ-uple of algorithms $a_1 \ldots a_\kappa$. Thanks to both

small κ values ($\kappa = 3$ in the experiments) and small number of algorithms (≤ 31 in the ICON challenge, section 4), the optimal κ-uple is determined by exhaustive search conditionally to the T_{ps} value.

The ASAP.V1 pre-scheduler finally relies on the 1-dimensional optimization of the overall computational budget T_{ps} allocated to the pre-scheduler. In all generality, the optimization of T_{ps} is a multi-objective optimization problem, e.g., balancing the overall number of problems solved and the overall computational budget. Multi-objective optimization commonly proceeds by determining the so-called Pareto front, made of non-dominated solutions. In our case, the Pareto front depicts how the performance varies with the overall computational budget, as illustrated on Fig. 2, where the performance is set to the number of solved instances.

In multi-objective decision making [2], the choice of a solution on the Pareto front is tackled using post-optimal techniques [4], including: i) compromise programming, where one wants to find the point the closest to an ideal target in the objective space; ii) aggregation of the objectives into a single one, e.g., using linear combination; or iii) marginal rate of return. The last heuristics consists of identifying the so-called "knees", that is, the points where any small improvement on a given criterion is obtained at the expense of a large decrease on another criterion, defining the so-called marginal rate of return. The vanilla marginal rate of return is however sensitive to strong local discontinuities; for instance, it would select point A in Fig. 2. Therefore, a variant taking into account the global shape of the curve, and measuring the marginal rate of improvement w.r.t. the extreme solutions on the Pareto front is used (e.g., selecting point K instead of point A in Fig.2).

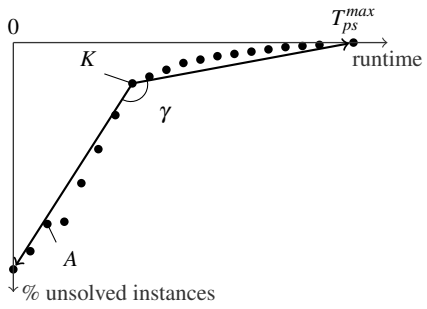

Fig. 2 Among a set of Pareto-optimal solutions, solution A has the best marginal rate of return; solution K, which maximizes the average rate of return w.r.t. the extreme solutions of the Pareto front (minimizing angle γ), is the knee selected in ASAP.

3.3 ASAP.V1 algorithm selector

As detailed in section 3.1, the AS relies on the performance model learned from the training problem instances. Two machine learning (ML) algorithms are considered in this paper: random forests and k-nearest neighbors (where the considered distance is the Euclidean distance in the description space of the problem instances). One hyper-parameter was adapted for each ML approach (all other hyper-parameters being set to their default value, using the Python scikit-learn library [18]), based on a few preliminary experiments: 35 trees are used for the RandomForest algorithm and the number of neighbors is set to $k = 3$ for the k-nearest neighbors. In the latter case, the predicted value associated to problem instance $\mathbf{x}$ is set to the weighted sum of the performance of its nearest neighbors, weighted by their relative distance to $\mathbf{x}$:

$$\widehat{\mathscr{G}}(\mathbf{x},a) = \frac{\sum_i ||\mathbf{x}-\mathbf{x}_i|| \mathscr{G}(a,\mathbf{x}_i)}{\sum_i ||\mathbf{x}-\mathbf{x}_i||}$$

where $\mathbf{x}_i$ ranges over the 3 nearest neighbors of $\mathbf{x}$. The description of the instances is normalized (each coordinate having zero mean and unit variance).

A main difficulty comes from the descriptive features forming the representation of problem instances. Typically, the feature values are missing for some groups of features, for quite a few problem instances, due to diverse causes (computation exceeded time limit, exceeded memory, presolved the instance, crashed, other, unknown). Missing feature values are handled by i) replacing the missing value by the feature average value; ii) adding to the set of descriptive features 7 additional boolean features per group of initial features, indicating whether the feature group values are available or the reason why they are missing otherwise.[1]

3.4 ASAP.V2

Several extensions of ASAP.V1 have been considered after the closing of the ICON challenge, aimed at exploring a richer pre-scheduler-AS search space while preventing the risk of overfitting induced by a larger search space.

We investigated the use of different time-outs for each algorithm in the pre-scheduler, while keeping the set of algorithms $(a_1,\ldots,a_\kappa)$ and the overall computational budget T_{ps}. The sequential optimization strategy (section 3.2), deterministically selecting T_{ps} as the solution with maximal average return rate, exhaustively determining the κ-uple of algorithms conditionally to T_{ps}, is thus extended to optimize the $(\tau_1,\ldots\tau_{\kappa-1})$ vector conditionally to $\sum_{i=1}^{\kappa-1}\tau_i \leq T_{ps}$, using a prominent continuous black-box optimizer, specifically the Covariance-Matrix Adaptation-Evolution Strategy (CMA-ES) [7].

[1] The increase in the overall number of features is handled by an embedded feature selection mechanism, removing all features with negligible importance criterion ($< 10^{-5}$ in the experiments) in a independenty learned 10-trees random forest regression model.

This extended search space is first investigated by considering the **raw optimization criterion** $\mathscr{F}_{raw}(\tau_1,\dots\tau_\kappa)$ measuring the cumulative performance of ASAP over all training problem instances, defined as follows:

$$\mathscr{F}_{raw}(\tau_1,\dots\tau_\kappa) = \sum_j Solv(\tau_1,\dots,\tau_\kappa,\mathbf{x}_j) \tag{3}$$

where $Solv(\tau_1,\dots,\tau_\kappa,\mathbf{x}_j)$ is the time required by the pre-scheduler $(a_i,\tau_i)_{i=1}^{\kappa}$ followed by AS_{init} to solve the j-th instance, or 10 times the dataset time-out.

However a richer search space entails some risk of overfitting, where the higher performance on data used to optimize ASAP (training data) is obtained at the expense of a lower performance on test data. Generally speaking, the datasets used to train an AS are small ones (a few hundred to a few thousand).

A **penalized optimization criterion** is thus considered:

$$\mathscr{F}_{L_2}\left((a_i,\tau_i)_{i=1}^{\kappa}\right) = \mathscr{F}\left((a_i,\tau_i)_{i=1}^{\kappa}\right) + w\sqrt{\sum_{i=1}^{\kappa}\left(\tau_i/T_{ps}\right)^2}$$

which penalizes uneven time sharing within the pre-scheduler (the regularization term is minimized when $\tau_i = \frac{1}{\kappa}T_{ps}$). The rationale for this penalization is to prevent brittle improvements on the training set due to opportunistic adjustments of the τ_is, at the expense of stable performances on further instances. The penalization weight w is adjusted using a preliminary cross-validation process.

A **randomized optimization criterion** is also considered. By construction, the ideal fitness function to be minimized is the expected performance over the problem domain. Only the empirical average performance over the problem instances is available, defining a noisy optimization problem. Sophisticated approaches have been proposed to address the noisy optimization issue (see e.g. [3]). Another approach is proposed here, based on the bootstrap principle: in each CMA-ES generation, the set of n problem instances used to compute the performance is uniformly drawn with replacement from the n-size training set. In this manner, each optimization generation considers a slightly different optimization objective noted $\mathscr{F}_{rand}$, thereby discouraging hazardous improvements and contributing to a more robust search.

Finally, a **probabilistic optimization criterion** is considered, handling the ASAP performance on a single problem instance as a random variable with a triangle-shape distribution (Fig. 3) centered on the actual performance $p(\mathbf{x})$, with support in $[p(\mathbf{x})-\theta, p(\mathbf{x})+\theta]$, and taking the expectation thereof. The merit of this triangular probability distribution function is to allow for an analytical computation of the overall fitness expectation, noted $\mathscr{F}_{dfp}$.

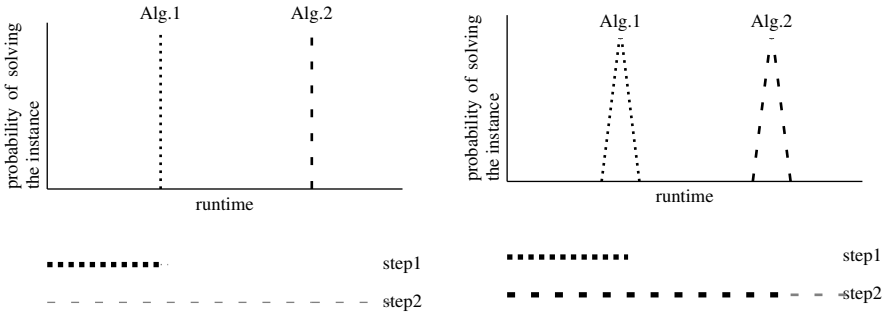

Fig. 3 Impact of a probabilistic optimization criterion: Difference between deterministic and probabilistic execution time. Left: the schedule deterministically stops as Alg. 1 solves the instance. Right: with some probability, Alg. 1 does not solve the instance and the execution proceeds.

4 Experimental setting: The ICON challenge

4.1 ASlib data format

Due to the difficulty of comparing the many algorithm selection systems and the high entry ticket to the AS field, a joint effort was undertaken to build the Algorithm Selection Library (ASlib), providing comprehensive resources to facilitate the design, sharing and comparison of AS systems [1]. ASlib (version 1.0.1) involves 13 datasets, also called scenarios (Table 1), gathered from recent challenges and surveys in the operations research, artificial intelligence and optimization fields. The interested reader is referred to [1] for a more comprehensive presentation.

Table 1 ASlib datasets (V1.0.1)

dataset	# instances	# algorithms	# features
ASP-POTASSCO	1294	11	138
CSP-2010	2024	2	86
MAXSAT12-PMS	876	6	37
PREMARSHALLING-ASTAR-2013	527	4	16
PROTEUS-2014	4021	22	198
QBF-2011	1368	5	46
SAT11-HAND	296	15	115
SAT11-INDU	300	18	115
SAT11-RAND	600	9	115
SAT12-ALL	1614	31	115
SAT12-HAND	767	31	115
SAT12-INDU	1167	31	115
SAT12-RAND	1362	31	115

Each dataset includes i) the performance and computation status of each algorithm on each problem instance; ii) the description of each problem instance, as a

vector of the expert-designed feature values (as said, this description considerably facilitates the comparison of the AS systems); iii) the computational status of each such feature (e.g., indicating whether the feature could be computed, or if it failed due to insufficient computational or memory resources). Last but not least, each dataset is equi-partitioned into 10 subsets, to enforce the reproducibility of the 10 fold cross-validation assessment of every AS algorithm.

4.2 The ICON Challenge on Algorithm Selection

The ICON Challenge on Algorithm Selection, within the ASlib framework, was carried on between February and July 2015 to evaluate AS systems in a fair, comprehensive and reproducible manner.[2] Each submitted system was assessed on the 13 ASlib datasets [1] with respect to three measures: i) number of problem instances solved; ii) extra runtime compared with the virtual best solver (VBS, also called oracle); and iii) Penalized Average Time-10 (PAR10) which is the cumulative runtime needed to solve all problem instances (set to ten times the overall computational budget whenever the problem instance is unsolved).

As the whole datasets were available to the community from the start, the evaluation was based on hidden splits between training and test set. Each submitted system provides a dataset-dependent, instance-dependent schedule of algorithms, optionally preceded by a dataset-dependent presolver (single algorithm running on all instances during a given runtime before the per-instance schedule runs). Each system can also, in a dataset-dependent manner, specify the groups of features to be used (in order to save the time needed to compute useless features).

Two baselines are considered: the oracle, selecting the best algorithm for each problem instance; and the single best (SB) algorithm, with best average performance over all problem instances in the dataset. The baselines are used to normalize every system performance over all datasets, associating performance 0 to the oracle (respectively performance 1 to the single best), supporting the aggregation of the system results over all datasets.

5 Experimental validation

5.1 Comparative results

Table 2 reports the results of all submitted systems on all datasets (the statistical significance tests are reported in Fig. 5). The general trend is that zilla algorithms dominate all other algorithms on the SAT datasets, as expected since they have con-

[2] The codes of all submitted systems and the results are publicly available, http://challenge.icon-fet.eu/challengeas

Table 2 Normalized performances of submitted systems, aggregated across all folds and all measures (the lower, the better). Ranks of zilla (challenge winner) and ASAP_RF.V1 (honourable mention) are given in parenthesis. Numbers were computed from the challenge outputs. Note: "PREMARSHALLING" stands for "PREMARSHALLING-ASTAR-2013".

	ASAP_RF.V1	ASAP_kNN.V1	autofolio	flexfolio	sunny	sunny-presolv	zilla	zillafolio
ASP-POTASSCO	0.294 (2)	0.359	0.299	0.314	0.37	0.336	0.319 (5)	**0.283**
CSP-2010	**0.146** (1)	0.247	0.288	0.223	0.263	0.406	0.2 (3)	0.157
MAXSAT12-PMS	0.168 (4)	0.159	0.45	**0.149**	0.166	0.224	0.201 (5)	0.233
PREMARSHALLING	0.349 (4)	0.369	0.359	0.307	0.325	**0.296**	0.374 (7)	0.385
PROTEUS-2014	0.16 (4)	0.177	0.222	**0.056**	0.134	0.103	0.245 (8)	0.223
QBF-2011	0.097 (2)	**0.091**	0.169	0.096	0.142	0.162	0.191 (7)	0.194
SAT11-HAND	0.341 (4)	0.318	0.342	0.342	0.466	0.464	0.328 (3)	**0.302**
SAT11-INDU	1.036 (5)	0.957	**0.875**	1.144	1.13	1.236	0.905 (2)	0.966
SAT11-RAND	0.104 (6)	0.09	**0.046**	0.226	0.116	0.088	0.053 (2)	0.067
SAT12-ALL	0.392 (5)	0.383	0.306	0.502	0.509	0.532	**0.273** (1)	0.322
SAT12-HAND	0.334 (5)	0.31	**0.256**	0.434	0.45	0.467	0.272 (2)	0.296
SAT12-INDU	0.955 (6)	0.919	0.604	0.884	1.074	1.018	0.618 (3)	**0.594**
SAT12-RAND	1.032 (5)	1.122	0.862	1.073	1.126	0.97	**0.779** (1)	0.79

sistently dominated the SAT contests in the last decade. On non-SAT problems however, zilla algorithms are dominated by ASAP_RF.V1.

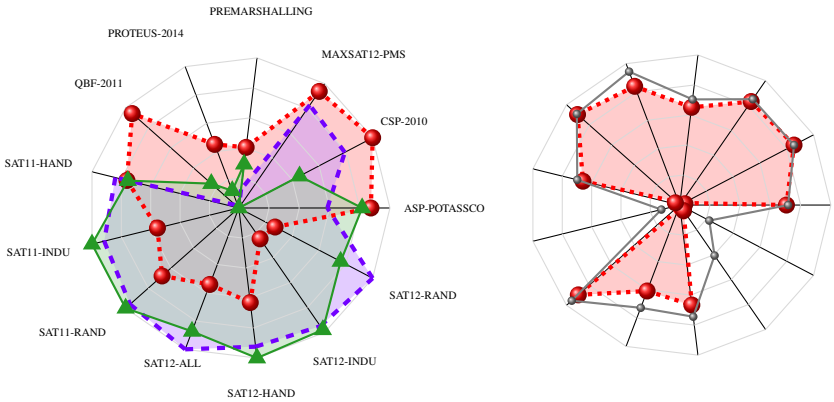

Fig. 4 Comparative performances. Left: per-dataset performances of ASAP_RF.V1 (balls, dotted line), zilla (no marker, dashed line) and autofolio (triangles, solid line); the scale is such that 0 corresponds to the worst submitted AS and 1 to the best submitted AS. Right: comparison of ASAP_RF.V1 and the best submitted AS per dataset in normalized performance (small balls, solid line). On all scenarios except 3, ASAP_RF reaches similar performances as the best submitted AS on this scenario.

The robustness of the ASAP approach is demonstrated as they never rank last; they however perform slightly worse than the single best on some datasets. The rescaled performances of ASAP_RF.V1 is compared to zilla and autofolio (Fig. 4, on the left), demonstrating that ASAP_RF.V1 offers a balanced performance, significantly lower than for zilla and autofolio on the SAT problems, but significantly higher on the other datasets; in this respect it defines an overall robust AS approach.

5.2 Sensitivity analysis

The sensitivity analysis conducted after the closing of the challenge compares ASAP.V2 (with different time-outs in the pre-scheduler) and ASAP.V1, and examines the impact of the different optimization criteria, aimed at avoiding overfitting: the raw fitness, the L_2-penalized fitness, the randomized fitness and the probabilistic fitness (Section 3.4).

The impact of the hyper-parameters used in the AS (number of trees set to 35, 100, 200, 300 and 500 trees in the Random Forest) is also investigated.

Table 3 summarizes the experimental results that each ASAP.V2 configuration would have obtained in the ICON challenge framework[3], together with the actual submissions results, including systems that were not competing in the challenge: llama-regr and llama-regrPairs from the organizers, and autofolio-48 which is identical to autofolio but with 48h time for training (12h was the time limit authorized in the challenge) [10].

The significance analysis, using a Wilcoxon signed-rank test, is reported in Fig. 5. A first result is that all ASAP.V2 variants improve on ASAP.V1 with significance level 1%. A second result is that ASAP.V2 with the probabilistic optimization criterion is not statistically significantly different from zilla, autofolio and zillafolio.

A third and most surprising result is that the difference between the challenge-winner zilla and most of ASAP.V2 variants is not statistically significant.

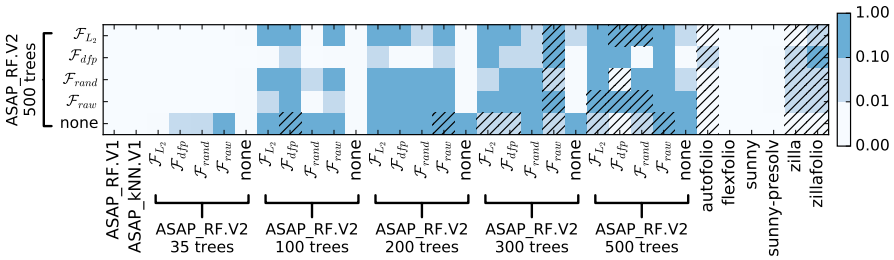

Fig. 5 Significance analysis after Wilcoxon signed-rank test p-value: ASAP_RF.V2(fitness variants, 500 trees) against all other systems. Color indicates the significance; hatched indicates that the line algorithm is outperformed by the column algorithm.

[3] For CSP-2010 dataset, only two algorithms are available: the pre-scheduler thus consists of a single algorithm, and all ASAP_RF.V2 variants with the same selector hyperparameter are identical.

Table 3 Optimized pre-scheduler performances aggregated across all datasets, all splits and all measures (the lower, the better). The hyperparameters for $\mathscr{F}_{L_2}$ and $\mathscr{F}_{dfp}$ were chosen after preliminary experiments using the cross validation provided with ASlib. For each configuration of the selector, the best-evaluated fitness function appears in bold.

fitness function (if relevant)	$\mathscr{F}_{L_2}$	$\mathscr{F}_{dfp}$	$\mathscr{F}_{rand}$	$\mathscr{F}_{raw}$	none
ASAP_RF.V2 35 trees	0.416	0.414	0.412	**0.410**	0.414
ASAP_RF.V2 100 trees	0.404	**0.398**	0.405	0.402	0.414
ASAP_RF.V2 200 trees	0.404	0.402	0.402	**0.399**	0.405
ASAP_RF.V2 300 trees	0.399	0.399	0.402	**0.393**	0.405
ASAP_RF.V2 500 trees	0.398	**0.394**	0.398	0.398	0.401
ASAP_RF.V1	0.416				
ASAP_kNN.V1	0.423				
autofolio	0.391				
flexfolio	0.442				
sunny	0.482				
sunny-presolv	0.485				
zilla	0.366				
zillafolio	0.37				
autofolio-48	0.375				
llama-regrPairs	0.395				
llama-regr	0.425				

equivalent to the means over the columns of Table 2

Fig. 6 details per dataset the performance improvement between ASAP.V2 (500 trees, $\mathscr{F}_{L_2}$ version) and ASAP.V2 (500 trees, $\mathscr{F}_{dfp}$ version) and on the other hand ASAP.V1_RF (35 trees). Note that ASAP.V2 outperforms the per-dataset best submission to the challenge for 3 datasets.

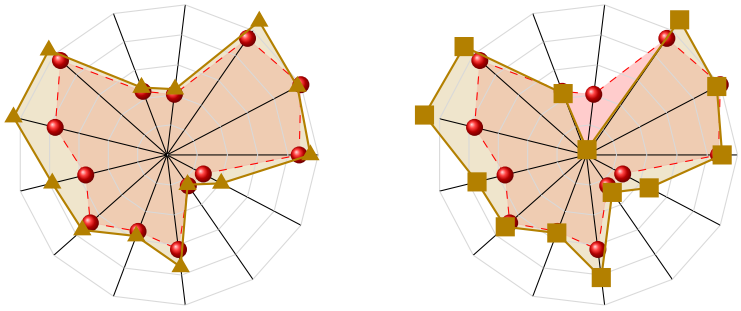

Fig. 6 Left: Comparison of ASAP_RF.V2 ($\mathscr{F}_{L_2}$, 500 trees, triangles) with ASAP_RF.V1. Right: Comparison of ASAP_RF.V2 ($\mathscr{F}_{dfp}$, 500 trees, squares) with ASAP_RF.V1.

5.3 ASAP.V2 optimized pre-scheduler behavior analysis

ASAP.V1 was designed after the observation that its pre-scheduler and AS components should be complementary. ASAP.V2 introduces a specific tuning of the pre-scheduler to strengthen the division of labor between them. Specifically, the optimized pre-scheduler should i) focus on trying to solve instances ill-handled by the selector and ii) be more efficient than its non-optimized counterpart to solve "easy" instances. The comparison of ASAP.V2 variants and the non-optimized pre-scheduler (rightmost column in Tab. 3, with equal time-outs) shows that these goals are met to some extent.

On the one hand, the pre-scheduler fine-tuning does improve the pre-scheduler performance; the overlap between instances that each component standalone can solve[4] (within T_{ps} for the pre-scheduler, within the remaining time for the AS) tends to diminish when optimizing the pre-scheduler for most datasets, as depicted on Fig. 7. On the other hand, the full ASAP.V2 systems (optimized pre-scheduler + AS) solve roughly as many instances as the non-optimized setup (difference $< 1\%$). It follows that the pre-scheduler fine-tuning leads to a better specialization of both components, though it does not translate into a global improvement.

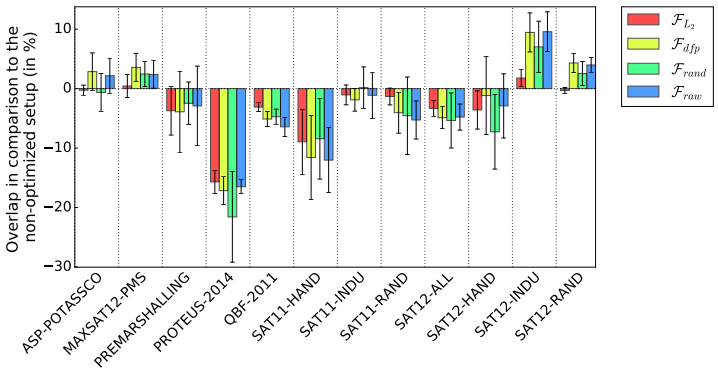

Fig. 7 Per-dataset number of instances that *can* be solved by both the pre-scheduler and the selector components in comparison to the non-optimized pre-scheduler variant.

The inclusion of T_{ps} in the set of optimized variables is expected to further strengthen this specialization.

The time spent in the pre-scheduling phase is reduced (up to 29%) by the pre-scheduler fine-tuning, as illustrated in Fig. 8. As one could have expected, the use of the L_2 regularization mitigates this effect (remind this setup prevents the optimized τ_is to be far apart from the equal time-outs of the the non-optimized pre-scheduler): it is low-risk, low-reward. No clear winner emerges from the other 3 variants.

[4] Remind that these instances are not actually passed to the AS in the challenge evaluation setup.

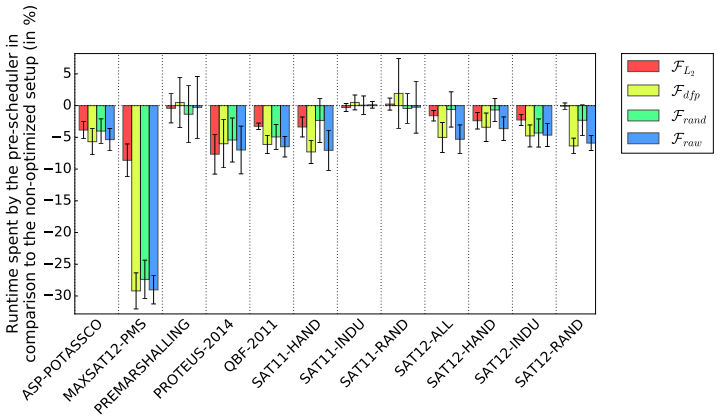

Fig. 8 Per-dataset runtime spent in the pre-scheduler phase in comparison to the non-optimized pre-scheduler variant.

6 Conclusion and perspectives

This paper presents two contributions. The first one is a new hybrid algorithm selection approach, the ASAP system, combining a pre-scheduler and a per-instance algorithm selector. ASAP.V1 introduces an algorithm selector learned conditionally to a predetermined schedule so that it focuses on instances that were not solved by the pre-scheduler. ASAP.V2 completes the loop as it re-optimizes the pre-scheduler conditionally to the new AS. A main lesson learned is that the scheduler and the AS must be optimized jointly to achieve an effective division of labor.

ASAP.V1, thoroughly evaluated in the ICON challenge on algorithm selection (ranked 4th) received an honourable mention, due to its novelty and good performance comparatively to the famed and long-known *zilla algorithms.

The second contribution is the ASAP.V2 extension, that achieves significantly better results along the same challenge setting. A main lesson learned is the importance of considering regularized optimization objectives: the amount of available data does not permit to consider richer AS search spaces without incurring a high risk of overfitting. In particular, the probabilistic performance criterion successfully contributed to a more stable optimization problem.

Further research will first comfort these good results by additional experiments on fresh data. A mid-term research is concerned with extending the optimization search space, and specifically adjusting the pre-scheduler parameters (number κ of algorithms and budget) depending on the scenario. Another perspective is concerned with learning the margin between any two algorithms depending on the problem instance, in order to yield a better AS, taking inspiration from [24]. The long term research will be devoted to move from alternate optimization of ASAP.V2 components, to a global optimization problem.

References

1. Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, M., Malitsky, Y., Fréchette, A., Hoos, H., Hutter, F., Leyton-Brown, K., Tierney, K., et al.: Aslib: A benchmark library for algorithm selection. Artificial Intelligence **237**, 41–58 (2016)
2. Branke, J., Deb, K., Dierolf, H., Osswald, M.: Finding knees in multi-objective optimization. In: Parallel Problem Solving from Nature-PPSN VIII, pp. 722–731. Springer (2004)
3. Cauwet, M., Liu, J., Rozière, B., Teytaud, O.: Algorithm portfolios for noisy optimization. Ann. Math. Artif. Intell. **76**(1-2), 143–172 (2016)
4. Deb, K.: Multi-objective evolutionary algorithms: Introducing bias among pareto-optimal solutions. In: Advances in evolutionary computing, pp. 263–292. Springer (2003)
5. Gagliolo, M., Schmidhuber, J.: Algorithm portfolio selection as a bandit problem with unbounded losses. Annals of Mathematics and Artificial Intelligence **61**(2), 49–86 (2011)
6. Gomes, C.P., Selman, B.: Algorithm portfolios. Artificial Intelligence **126**(1), 43–62 (2001)
7. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized Evolution Strategy with Covariance Matrix Adaptation. Evolutionary Computation **11**(1), 1–18 (2003)
8. Huberman, B.A., Lukose, R.M., Hogg, T.: An economics approach to hard computational problems. Science **275**(5296), 51–54 (1997)
9. Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm selection and scheduling. In: Proc. 17th CP, pp. 454–469. LNCS 6876, Springer (2011)
10. Kotthoff, L.: ICON challenge on algorithm selection. CoRR **abs/1511.04326** (2015)
11. Kotthoff, L.: Algorithm Selection for Combinatorial Search Problems: A Survey, pp. 149–190. Springer International Publishing (2016)
12. Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., Shoham, Y.: A portfolio approach to algorithm selection. In: Proc. IJCAI, pp. 1542–1543 (2003)
13. Lindauer, M., Bergdoll, R.D., Hutter, F.: An empirical study of per-instance algorithm scheduling. In: Proc. LION10, pp. 253–259. Springer (2016)
14. Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm portfolios based on cost-sensitive hierarchical clustering. In: Proc. 23rd IJCAI, pp. 608–614. AAAI Press (2013)
15. Mısır, M., Sebag, M.: Alors: An algorithm recommender system. Artificial Intelligence **244**, 291–314 (2017). Published on-line Dec. 2016
16. O´Mahony, E., Hebrard, E., Holland, A., Nugent, C., O´Sullivan, B.: Using case-based reasoning in an algorithm portfolio for constraint solving. In: Proc. ICAICS, pp. 210–216 (2008)
17. Oentaryo, R.J., Handoko, S.D., Lau, H.C.: Algorithm selection via ranking. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI) (2015)
18. Pedregosa, F. et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
19. Rice, J.R.: The algorithm selection problem. Advances in Computers **15**, 65–118 (1976)
20. Stern, D., Herbrich, R., Graepel, T., Samulowitz, H., Pulina, L., Tacchella, A.: Collaborative expert portfolio management. In: Proc. 24th AAAI, pp. 179–184 (2010)
21. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997)
22. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: Features for SAT (2012). University of British Columbia
23. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Satzilla: portfolio-based algorithm selection for sat. Journal of Artificial Intelligence Research pp. 565–606 (2008)
24. Xu, L., Hutter, F., Shen, J., Hoos, H.H., Leyton-Brown, K.: Satzilla2012: Improved algorithm selection based on cost-sensitive classification models. Proc. SAT Challenge pp. 57–58 (2012)
25. Yun, X., Epstein, S.L.: Learning algorithm portfolios for parallel execution. In: Y. Hamadi, M. Schoenauer (eds.) Proc LION 6, pp. 323–338. LNCS 7219, Springer (2012)