SDU
University of Southern Denmark

On the Evaluation of (Meta-)solver Approaches

Amadini, Roberto; Gabbrielli, Maurizio; Liu, Tong; Mauro, Jacopo

Published in:
Journal of Artificial Intelligence Research

DOI:
10.1613/JAIR.1.14102

Publication date:
2023

Document version:
Final published version

Citation for pulished version (APA):
Amadini, R., Gabbrielli, M., Liu, T., & Mauro, J. (2023). On the Evaluation of (Meta-)solver Approaches. *Journal of Artificial Intelligence Research*, *76*, 705-719. https://doi.org/10.1613/JAIR.1.14102

Go to publication entry in University of Southern Denmark's Research Portal

Terms of use
This work is brought to you by the University of Southern Denmark.
Unless otherwise specified it has been shared according to the terms for self-archiving.
If no other license is stated, these terms apply:

- You may download this work for personal use only.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying this open access version

If you believe that this document breaches copyright please contact us providing details and we will investigate your claim.
Please direct all enquiries to puresupport@bib.sdu.dk

Download date: 11. Nov. 2024

Journal of Artificial Intelligence Research 76 (2023) 705-719 Submitted 07/2022; published 03/2023

Research Note

On the Evaluation of (Meta-)solver Approaches

Roberto Amadini ROBERTO.AMADINI@UNIBO.IT
Maurizio Gabbrielli MAURIZIO.GABBRIELLI@UNIBO.IT
Department of Computer Science and Engineering,
University of Bologna, Italy

Tong Liu LTEU@ICLOUD.COM
Meituan,
Beijing, China

Jacopo Mauro MAURO@IMADA.SDU.DK
Department of Mathematics and Computer Science,
University of Southern Denmark, Denmark

Abstract

Meta-solver approaches exploit many individual solvers to potentially build a better solver. To assess the performance of meta-solvers, one can adopt the metrics typically used for individual solvers (e.g., runtime or solution quality) or employ more specific evaluation metrics (e.g., by measuring how close the meta-solver gets to its virtual best performance). In this paper, based on some recently published works, we provide an overview of different performance metrics for evaluating (meta-)solvers by exposing their strengths and weaknesses.

1. Introduction

A famous quote attributed to Aristotle says that "*the whole is greater than the sum of its parts*". This principle has been applied in several contexts, including the field of constraint solving and optimization. Combinatorial problems arising from application domains such as scheduling, manufacturing, routing or logistics can be tackled by combining and leveraging the complementary strengths of different solvers to create a better global *meta-solver*.[1]

Several approaches for combining solvers and hence creating effective meta-solvers have been developed. Over the last decades, we witnessed the creation of new Algorithm Selection (AS) (Kotthoff, 2016) and Configuration (H. H. Hoos, 2012) approaches[2] that reached peak results in various solving competitions (SAT competition, 2021; Stuckey, Becket, & Fischer, 2010; ICAPS, 2021). To compare different meta-solvers, new competitions were created, e.g., the 2015 ICON challenge (Kotthoff, Hurley, & O'Sullivan, 2017) and the 2017 OASC competition on algorithm selection (Lindauer, van Rijn, & Kotthoff, 2019). However, the discussion of why a particular evaluation metric has been chosen to rank the solvers is absent.

We believe that further study on comparing meta-solvers is necessary because meta-solvers are often evaluated on *heterogeneous* scenarios, characterized by different problems

1. Meta-solvers are sometimes referred in the literature as *portfolio solvers* because they take advantage of a "portfolio" of different solvers.
2. A fine-tuned solver can be seen as a meta-solver where we consider different configurations of the same solver as different solvers.

©2023 AI Access Foundation. All rights reserved.

having specific features, different timeouts, and different individual solvers from which the meta-solver approaches are built. From our point of view, this is one of the main differences w.r.t. the evaluation of individual solvers. Indeed, the latter are typically assessed in more homogeneous scenarios, where a fixed number of solvers compete against each other on related problems (e.g., SAT problems, ASP problems, Planning problems, etc.) within a fixed time window. In this work, we will consider the scenarios of the Algorithm Selection library (ASlib) (Bischl et al., 2016), i.e., the reference library for AS scenarios. The ASlib contains several data sets from the literature, and we assume that they represent realistic scenarios.

Starting from some surprising results presented by Liu, Amadini, Mauro, and Gabbrielli (2021) showing dramatic ranking changes with different but reasonable metrics, we would like to draw more attention to the evaluation of meta-solver approaches by shedding some light on the strengths and weaknesses of different metrics. Unsurprisingly, some of the findings we report here also apply to the evaluation of individual solvers.

2. Evaluation Metrics

Before talking about the evaluation metrics, we should spend some words on what we need to evaluate: the solvers. In our context, a solver is a program that takes as input the description of a computational problem in a given language and returns an observable *outcome* providing zero or more solutions for the given problem. For example, for decision problems, the outcome may be simply "yes" or "no" while for optimization problems, we might be interested in the best solutions found along the search. An *evaluation metric*, or performance metric, is a function mapping the outcome of a solver on a given instance to a number representing "how good" the solver is on this instance.

An evaluation metric is often not just defined by the output of the (meta-)solver. Indeed, it can be influenced by other actors, such as the computational resources available, the problems on which we evaluate the solver, and the other solvers involved in the evaluation. For example, it is often unavoidable to set a *timeout* τ on the solver's execution when there is no guarantee of termination in a reasonable amount of time (e.g., NP-hard problems). Timeouts make the evaluation feasible but inevitably couple the evaluation metric to the execution context. For this reason, the evaluation of a meta-solver should also consider the *scenario* that encompasses the solvers to evaluate, the instances used for the validation, and the timeout. Formally, at least for the purposes of this paper, we can define a scenario as a triple $(\mathcal{I}, \mathcal{S}, \tau)$ where: $\mathcal{I}$ is a set of problem instances, $\mathcal{S}$ is a set of *individual* solvers, $\tau \in (0, +\infty)$ is a timeout such that the outcome of solver $s \in \mathcal{S}$ over instance $i \in \mathcal{I}$ is always measured in the time interval $[0, \tau)$.

Evaluating meta-solvers over heterogeneous scenarios $(\mathcal{I}_1, \mathcal{S}_1, \tau_1), (\mathcal{I}_2, \mathcal{S}_2, \tau_2), \ldots$, is complicated by the fact that the sets of instances $\mathcal{I}_k$, the sets of solvers $\mathcal{S}_k$ and the timeouts τ_k can be very different. As we shall see in Sect. 2.3, things are even trickier in scenarios including optimization problems.

2.1 Absolute vs Relative Metrics

A sharp distinction between evaluation metrics can be drawn depending on whether their values depend on the outcome of other solvers. We say that an evaluation metric is *relative*

in the former case, *absolute* otherwise. For example, a well-known absolute metric is the *penalized average runtime* with penalty $\lambda \geq 1$ (PAR_λ) that compares the solvers by using the average solving runtime and penalizes the timeouts with λ times the timeout.

Formally, let $\texttt{time}(i, s, \tau)$ be the function returning the runtime of solver s on instance i with timeout τ, assuming $\texttt{time}(i, s, \tau) = \tau$ if s cannot solve i before the timeout τ. For optimization problems, we consider the runtime as the time taken by s to solve i to optimality[3] assuming w.l.o.g. that an optimization problem is always a minimization problem. We can define PAR_λ as follows.

Definition 1 (Penalized Average Runtime)**.** *Let $(\mathcal{I}, \mathcal{S}, \tau)$ be a scenario, the PAR_λ score of solver $s \in \mathcal{S}$ over $\mathcal{I}$ is given by $\frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \texttt{par}_\lambda(i, s, \tau)$ where:*

$$\texttt{par}_\lambda(i, s, \tau) = \begin{cases} \texttt{time}(i, s, \tau) & \text{if } \texttt{time}(i, s, \tau) < \tau \\ \lambda \cdot \tau & \text{otherwise.} \end{cases}$$

Well-known PAR measures are, e.g., the PAR_2 adopted in the SAT competitions (SAT competition, 2021) or the PAR_{10} used by Lindauer et al. (2019); Luo, Vallati, and Hoos (2019). The PAR score readily applies to meta-solvers: whether s is an individual solver does not affect the definition of PAR_λ (and, in general, of any absolute metric). However, it is important to note that meta-solvers are often evaluated over scenarios with different timeouts. This should imply a *normalization* of PAR_λ in a fixed range to avoid misleading comparisons due to the impact of different penalization when $\texttt{time}(i, s, \tau) = \tau$.

Another absolute metric for decision problems is the number (or percentage) of instances solved where ties are broken by favoring the solver minimizing the average running time, i.e., minimizing the PAR_1 score. This metric has been used in various tracks of the planning competition (ICAPS, 2021), the XCSP competition (XCSP Competition, 2019), and the QBF evaluations (QBFEVAL, 2021).

A well-established relative metric is instead the *Borda count*, adopted, e.g., by the MiniZinc Challenge (Stuckey, Feydy, Schutt, Tack, & Fischer, 2014) for both single solvers and meta-solvers. The Borda count is a family of voting rules applicable to the evaluation of a solver by considering the comparison as an election where the solvers are the candidates, and the problem instances are the voters. The MiniZinc challenge uses a variant of Borda[4] where each solver scores points proportionally to the number of solvers it beats. Assuming that $\texttt{obj}(i, s, t)$ is the best objective value found by solver s on optimization problem i at time t, with $\texttt{obj}(i, s, t) = +\infty$ when no solution is found at time t, the MiniZinc challenge score is defined as follows.

Definition 2 (MiniZinc challenge score)**.** *Let $(\mathcal{S}, \mathcal{I}, \tau)$ be a scenario where $\mathcal{I} = \mathcal{I}_{dec} \cup \mathcal{I}_{opt}$ with $\mathcal{I}_{dec}$ decision problems and $\mathcal{I}_{opt}$ optimization problems. The MiniZinc challenge*

3. If s cannot solve i to optimality before τ, then $\texttt{time}(i, s) = \tau$ even if sub-optimal solutions are found.
4. In the original definition, the lowest-ranked candidate gets 0 points, the next-lowest 1 point, and so on.

(MZNC) score of $s \in \mathcal{S}$ *over* $\mathcal{I}$ *is* $\sum_{i \in \mathcal{I}, s' \in S \setminus \{s\}} m_s(i, s', \tau)$ *where:*

$$m_s(i,s',\tau) = \begin{cases} 0 & \text{if } \texttt{unknown}(i,s,\tau) \vee \texttt{better}(i,s',s,\tau) \\ 1 & \text{if } \texttt{better}(i,s,s',\tau) \\ 0.5 & \text{if } \texttt{time}(i,s,\tau) = \texttt{time}(i,s',\tau) \\ & \text{and } \texttt{obj}(i,s,\tau) = \texttt{obj}(i,s',\tau) \\ \dfrac{\texttt{time}(i,s',\tau)}{\texttt{time}(i,s,\tau) + \texttt{time}(i,s',\tau)} & \text{otherwise} \end{cases}$$

where the predicate $\texttt{unknown}(i,s,\tau)$ *holds if* s *does not produce any solution within the timeout:*

$$\texttt{unknown}(i,s,\tau) = (i \in \mathcal{I}_{dec} \wedge \texttt{time}(i,s,\tau) = \tau) \vee (i \in \mathcal{I}_{opt} \wedge \texttt{obj}(i,s,\tau) = \infty)$$

and $\texttt{better}(i,s,s',\tau)$ *holds if* s *finishes earlier than* s' *or it produces a better solution:*

$$\texttt{better}(i,s,s',\tau) = (\texttt{time}(i,s,\tau) < \texttt{time}(i,s',\tau) \wedge \texttt{time}(i,s',\tau) = \tau) \vee (\texttt{obj}(i,s,\tau) < \texttt{obj}(i,s',\tau))$$

This is clearly a relative metric because changing the set of available solvers can affect the MiniZinc scores.

A relative and meta-solver-specific measure, adopted in the 2015 ICON and 2017 OASC challenges (Lindauer et al., 2019) to handle the disparate nature of the scenarios, is the *closed gap score*. This metric assigns to a meta-solver a value in $(-\infty, 1]$ proportional to how much it closes the gap between the best individual solver available, or *single best solver* (*SBS*), and the *virtual best solver* (*VBS*), i.e., an oracle-like meta-solver always selecting the best individual solver. The closed gap is actually a "meta-metric", defined in terms of another evaluation metric m to minimize. Formally, if $(\mathcal{I}, \mathcal{S}, \tau)$ is a scenario then $m(i, VBS, \tau) = \min\{m(i,s,\tau) \mid s \in \mathcal{S}\}$ for each $i \in \mathcal{I}$ and $SBS = \operatorname{argmin}_{s \in \mathcal{S}} \sum_{i \in \mathcal{I}} m(i,s,\tau)$. With these definitions

Definition 3 (Closed gap)**.** *Let* $(\mathcal{I}, \mathcal{S}, \tau)$ *be a scenario and* $m : \mathcal{I} \times (\mathcal{S} \cup \{S, VBS\}) \times [0, \tau] \to \mathbb{R}$ *an evaluation metric to minimize for that scenario, where* S *is a meta-solver over the solvers of* $\mathcal{S}$*. Let* $m_\sigma = \sum_{i \in \mathcal{I}} m(i, \sigma, \tau)$ *for* $\sigma \in \{S, SBS, VBS\}$*. The* closed gap *of* S *w.r.t.* m *on that scenario is:*[5]

$$\frac{m_{SBS} - m_S}{m_{SBS} - m_{VBS}}$$

The obvious assumption here is $m_{SBS} > m_{VBS}$, i.e., no single-solver can be the *VBS* (otherwise, no AS would be needed). Note that, unlike the PAR and MZNC scores, the closed gap is specifically tailored for meta-solvers. Indeed, applying it to individual solvers means assigning the score 0 to the *SBS* and a negative score to the other solvers linearly proportional to their performance difference w.r.t. the *SBS* and the gap $m_{SBS} - m_{VBS}$. This clearly makes little sense for individual solvers.

If not specified, we will assume that the closed gaps are computed w.r.t. the PAR_{10} score as done in the AS challenges 2015 and 2017.[6]

5. The definition of m is overloaded to be consistent with the notation that one can find, e.g., in (Lindauer et al., 2019).
6. In the 2015 edition, the closed gap was computed as $1 - \frac{m_{SBS} - m_S}{m_{SBS} - m_{VBS}} = \frac{m_{VBS} - m_S}{m_{VBS} - m_{SBS}}$.

Table 1: Comparison ASAP vs RF. The MZNC column reports the average MZNC score per scenario. Negative scores are in bold font.

	Closed gap		MZNC		Better than other	
Scenario	ASAP	RF	ASAP	RF	ASAP	RF
ASP-POTASSCO	0.7444	0.5314	2.2235	2.6163	275	671
BNSL-2016	0.8463	0.7451	1.2830	3.0250	98	993
CPMP-2015	0.6323	0.1732	2.0501	2.3660	137	334
CSP-MiniZinc-Time-2016	0.6251	0.2723	2.1552	2.7214	17	53
GLUHACK-2018	0.4663	0.4057	1.9040	2.4528	62	147
GRAPHS-2015	0.758	**-0.6412**	2.3045	3.3731	489	3663
MAXSAT-PMS-2016	0.5734	0.3263	1.4747	2.8616	66	439
MAXSAT-WPMS-2016	0.7736	**-1.1826**	1.5168	2.4043	126	386
MAXSAT19-UCMS	0.6583	**-0.2413**	2.0893	2.5189	145	269
MIP-2016	0.35	**-0.3626**	2.4035	2.4239	81	105
QBF-2016	0.7568	**-0.1366**	1.8642	2.7154	193	467
SAT03-16_INDU	0.3997	0.1503	2.1508	2.5812	491	1116
SAT12-ALL	0.7617	0.6528	1.6785	2.8250	262	1227
SAT18-EXP	0.5576	0.3202	1.9239	2.4998	61	164
TSP-LION2015	0.4042	**-19.1569**	2.4352	2.6979	1115	1949
Tot.	9.3077	**-18.1438**	29.4777	40.0622	3618	11983
Tot.−TSP-LION2015	8.9036	1.0131	27.0425	37.3642	2503	10034

2.2 A Surprising Outcome

An interesting outcome reported in Liu et al. (2021) was the profound difference between the closed gap and the MiniZinc challenge scores. Liu et al. compared the performance of six meta-solver approaches across 15 decision-problem scenarios taken from ASlib (Bischl et al., 2016) and coming from heterogeneous domains such as Answer-Set Programming, Constraint Programming, Quantified Boolean Formula, Boolean Satisfiability.

Tab. 1 reports the performance of meta-solvers ASAP and RF,[7] respectively the best approach according to the closed gap score and the MZNC score. The scores in the four leftmost columns clearly show a remarkable difference in rank if we swap the evaluation metric. With the closed gap, ASAP is the best approach and RF the *worst* among all the meta-solvers considered, while with the MZNC score RF climbs to the first position while ASAP drops to the *last* position. The fact that different metrics used in international competitions give diametrically opposite results should give pause for thought.

Another thing that catches the eye in Tab. 1 is the presence of *negative scores*. This happens because, by definition, the closed gap has upper bound 1 (no meta-solver can improve the *VBS*) but not a fixed lower bound. Hence, when the performance of the meta-solver is worse than the performance of the single best solver, the closed gap drops below zero. While, at first glance, this seems reasonable—meta-solvers should perform no worse than the individual solvers—it is worth noting that the penalty for performing worse than the SBS also depends on the denominator $m_{SBS} - m_{VBS}$. This means that in scenarios

7. RF uses a Random Forest classifier to predict the best solver of the portfolio for a given instance.

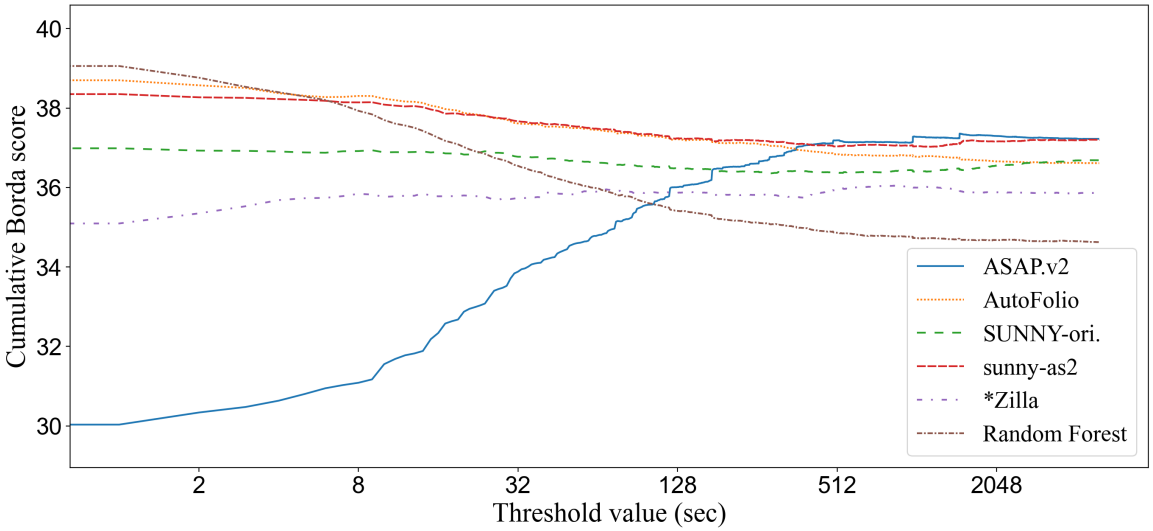

Figure 1: Cumulative Borda count by varying the δ threshold.

where the performance of the *SBS* is close to the perfect performance of the *VBS*, this penalty can be significantly magnified. The TSP-LION2015 scenario is a clear example: the RF approach gets a penalization of more than 19 points, i.e., more than 19 times the perfect score of the virtual best solver. This means that RF should perform flawlessly in about 20 other scenarios to compensate for this punishment. In fact, in TSP-LION2015 the PAR_{10} distributions of *SBS* and *VBS* are very close: the *SBS* is able to solve 99.65% of the instances solved by the *VBS*, leaving little room for improvement. RF scores -19.1569 while still solving more than 90% of the instances of the scenario and having a difference with ASAP of slightly more than 5% instances solved. Furthermore, in this scenario, RF beats ASAP on 1949 instances, while ASAP beats RF on 834 instances less (26.85% of the dataset).

Why are the closed gap and the MZNC rankings so different? Looking at the rightmost two columns in Tab. 1 showing, for each scenario, the number of instances where one approach is faster than the other, one may conclude that RF is far better than ASAP. Indeed, for *all* the scenarios, the number of instances where RF is faster than ASAP is bigger than the number of instances where ASAP is faster than RF. Overall, it is quite impressive that RF beats ASAP on 11983 instances while ASAP beats RF on 3618 times only.

An initial clue of why this happens is revealed in Liu et al. (2021), where a parametric version of MZNC score is used. Def. 2 is generalized by assuming the performance of two solvers equivalent if their runtime difference is below a given time threshold δ.[8] This variant was considered because a time difference of a few seconds could be considered irrelevant if solving a problem can take minutes or hours. The parametric MZNC score is depicted in Fig. 1, where different thresholds δ are considered on the x-axis. It is easy to see how the performance of ASAP and RF reverses when δ increases: ASAP bubbles from the bottom to the top, while RF gradually sinks to the bottom.

8. Formally, if $|\mathtt{time}(i,s,\tau) - \mathtt{time}(i,s',\tau)| \leq \delta$ then both s and s' scores 0.5 points. Note that if $\delta = 0$ we get the original MZNC score as in Def. 2.

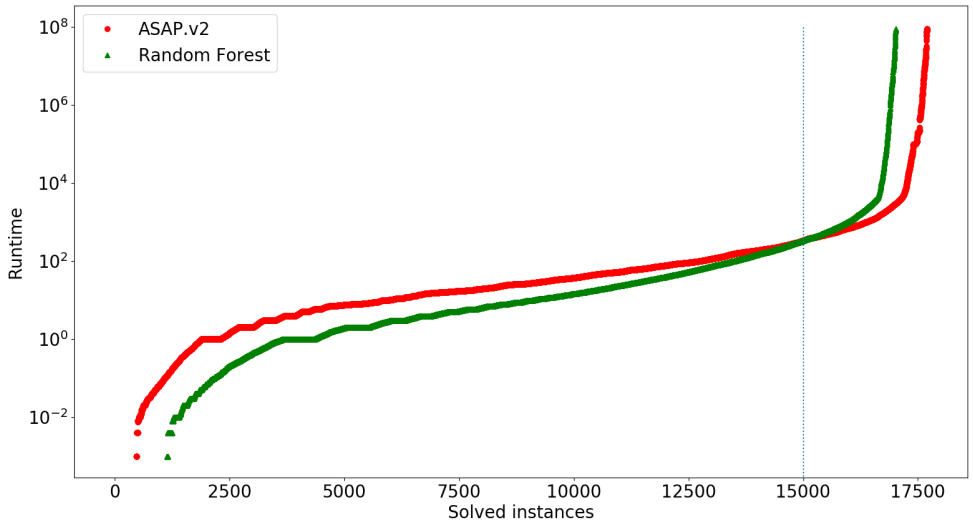

Figure 2: Runtime distribution of solved instances between ASAP and RF.

Let us further investigate this anomaly. Fig. 2 shows the runtime distributions of the instances solved by ASAP and RF, sorted by ascending runtime. We can see that ASAP solves more instances, but for around 15k instances, RF is never slower than ASAP. In summary, ASAP solves more instances, but RF generally is quicker when it solves an (often easy) instance. This entails the significant difference between the closed gap and Borda metrics.

In our opinion, on the one hand, it is fair to think that ASAP performs better than RF in these scenarios. The MZNC score seems to over-penalize ASAP w.r.t. RF. Moreover, from Fig. 1 we can also note that for $\delta \leq 103$ the parametric MZNC score of RF is still better, but 103 seconds looks quite a high threshold to consider two performances as equivalent. On the other hand, the closed gap score can also be over-penalizing due to negative outliers. The simplest fix for this issue would be to set the metric's lower bound to a parameter $\ell \leq 0$, but this would not discriminate between performances with a closed gap less than ℓ. Alternatively, one can modify the closed gap (c.f. Def. 3) when $m_s \geq m_{SBS}$ by using the "virtual *worst* solver" to establish a lower bound, thus assigning the score $\frac{m_{SBS} - m_s}{m_{VWS} - m_{VBS}}$ where $m_{VWS} = \sum_{i \in \mathcal{I}} \max\{m(i, s, \tau) \mid s \in \mathcal{S}\}$ is the performance of the "virtual *worst* solver". In this way, the closed gap will always be in $[-1, 1]$, with -1 denoting the worst possible performance, 0 the *SBS* performance and 1 the *VBS* performance.

We also point out that the definitions of *SBS* found in the literature do not clarify how it is computed in scenarios where the set of instances $\mathcal{I}$ is split into test and training sets. Should the *SBS* be computed on the instances of the training set, the test set, or the whole dataset $\mathcal{I}$? One would be inclined to use the test set to select the *SBS*, but this choice might be problematic because the test set is usually relatively small w.r.t. the training set when using, e.g., cross-validation methods. In this case, issues with negative outliers might

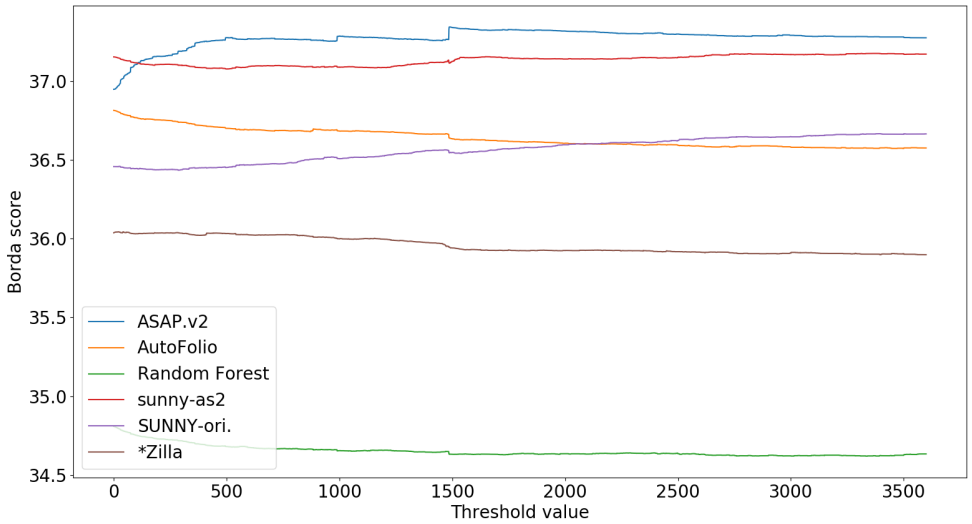

Figure 3: Modified Borda count by varying the δ threshold.

be amplified. If not clarified, this could lead to confusion. For example, in the 2015 ICON challenge, the *SBS* was computed by considering the entire dataset (training and testing instances together). In the 2017 OASC, instead, the *SBS* was originally computed on the test set of the scenarios, but later the results were amended by computing the *SBS* on the training set.

A possible way around the above issues of negative scores and amplification of small performance differences is to use a modified version of the MZNC score, as proposed in (Amadini, Gabbrielli, & Mauro, 2015) for individual solvers, where a meta-solver s gets a score of $0.5 + \dfrac{\texttt{time}(i, s', \tau) - \texttt{time}(i, s, \tau)}{2\tau}$ when the outcomes of s and any other meta-solver s' are indistinguishable, i.e., if they produce the same solution and their runtime difference does not exceed a given threshold δ. Fig. 3 shows the scores of the meta-solvers reported in Fig. 1 with this new definition, by varying $\delta \in [0, 3600]$. The difference between Fig. 1 and Fig. 3 is clear: with the modified score, the meta-solvers' ranking is more stable, especially for ASAP and RF. This can be explained with a simple example: suppose that $\tau = 1000$, $\texttt{time}(i, s, \tau) = 3$, and $\texttt{time}(i, s', \tau) = 9$ with $\delta = 5$. With the original MZNC score (Def. 2), s scores 0.75 while s' scores 0.25; with the modified one s scores 0.503 while s' scores 0.497. However, if $\texttt{time}(i, s, \tau) = 300$, and $\texttt{time}(i, s', \tau) = 900$ with the original MZNC score s and s' would achieve the same score, despite an absolute time difference of 6 and 600 seconds respectively. Instead, with the modified score s and s' would score 0.8 and 0.2, respectively. Alternatively, δ can be defined according to the percentage gap between the two solvers, i.e., two solvers are considered indistinguishable if their runtime difference is within 1% of the better one.

Another plausible alternative is to consider the *speedup* of a meta-solver w.r.t. the *SBS* or the *VBS*, i.e., how much a meta-solver can improve a baseline solver. This metric was

Table 2: Average closed gap, speedup, incomplete score and Relative Marginal Contribution (RMC). Peak performance in bold font.

Meta-solver	Closed gap	Speedup	Incomplete	RMC [%]
ASAP	**0.6205**	**0.1137**	0.4619	11.4606
sunny-as2	0.5126	0.1061	0.6591	12.6824
SUNNY-original	0.4893	0.0878	0.6303	8.5392
autofolio	0.4686	0.0927	0.6755	10.1789
RF-reg	0.4090	0.0842	0.6589	**28.3729**
*Zilla	0.2169	0.0734	0.5770	12.3025
RF	-1.2096	0.0492	**0.7108**	16.4635

mentioned in (Liu et al., 2021), inspired by what in (Lindauer et al., 2019) is called "improvement factor". We can compute the speedup of a meta-solver s as $\frac{m_{VBS}}{m_s}$ where m is a performance metric to be minimized. This is a normalized runtime measure, where m_s is scaled via m_{VBS} hence avoiding issues with scenarios having different timeouts. Unlike the closed gap, which has no lower bound, the speedup always falls in $(0, 1]$ with bigger values meaning better performance. An apparently similar approach is computing the speedup on a *per-instance* basis, i.e., considering $\frac{m(i, VBS, \tau)}{m(i, s, \tau)}$ for each instance $i \in \mathcal{I}$ of the scenario. Because m could be 0, one might use instead the incomplete score used in the MaxSAT Evaluation (*MaxSAT Evaluation*, 2022), i.e., $\sum_{i \in \mathcal{I}} \frac{1 + m(i, VBS, \tau)}{1 + m(i, s, \tau)}$ where m is the cost (i.e., the number of violated clauses in the context of MaxSAT).

Tab. 2 reports, using the data of (Liu et al., 2021) and $m = \mathsf{PAR}_{10}$, the average closed gap, speedup, and MaxSAT incomplete score for different meta-solvers. As additional baseline, we added a *regression* version of RF, called RF-reg, which uses RF with regression instead of classification to estimate the PAR10 of a solver for a new instance. RF-reg picks the solver of the portfolio with lowest predicted PAR10. Closed gap and speedup have similar definitions, with the main difference that the speedup does not take into account the *SBS* performance and does not allow negative scores. Therefore, it is not surprising that the closed gap and speedup results are similar, apart from SUNNY-original and autofolio, which swap their position, and the RF score, which is now positive and not so far from the other approaches. Instead, with the incomplete score ASAP and RF revert their positions similarly to what happens with the MZNC score. This happens because these two metrics have a similar approach: for each instance, the MZNC compares each solver against each other, while the incomplete score compares each solver against the best one. Note that Tab. 2 reports an adaptation of the MaxSAT incomplete score to PAR_{10}. However, this metric makes much more sense in its native context, i.e., for optimization problems (as we shall see in Sect. 2.3). RF-reg is not competitive w.r.t. to the best approaches in terms of closed gap, speedup, and incomplete score even though it is specifically trained with the PAR10 metric. If we look at Fig. 4, showing the runtime distributions of the instances solved by ASAP and RF-reg, we observe a trend similar to Fig. 2 where ASAP and RF are compared.

We believe that somehow a balance has to be found between the metrics of Tab. 2 when we evaluate meta-solvers. We should not over-reward a meta-solver that, after all, is the one

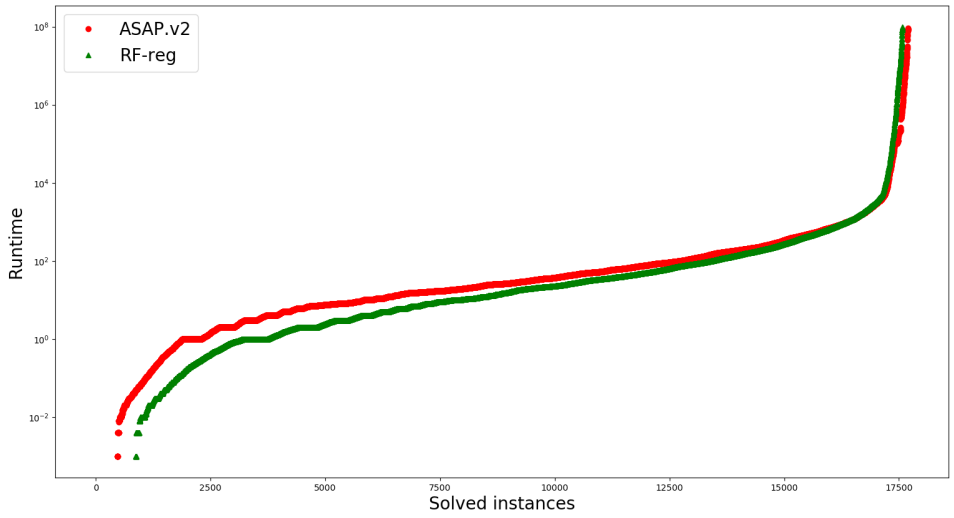

Figure 4: Runtime distribution of solved instances between ASAP and RF-reg.

with the lowest number of solved instances. On the one hand, we should not over-penalize slight time differences, which we believe are acceptable for meta-solvers. For example, it is natural for a meta-solver to spend some time in tasks such as pre-solving, feature selection, or parallel computation. Also, for the speedup the choice of λ matters if $m = \mathsf{PAR}_\lambda$. For example, with PAR_1 the best speedup approach would be sunny-as2 followed by autofolio, while ASAP would drop from the first to the third position.

The rightmost column of Tab. 2 reports instead the average *relative marginal contribution* (RMC) of each meta-solver over the different scenarios. This metric was used in the Sparkle SAT Challenge 2018 (Luo, Vallati, & Hoos, 2018) and the Sparkle Planning Challenge 2019 (Luo et al., 2019) to rank individual solvers (i.e., planners) according to the magnitude of their contribution to the VBS over the portfolio of all the participants. For each scenario, if $\mathcal{S}$ is the set of meta-solvers we consider, the RMC of $s \in \mathcal{S}$ over the instances of $\mathcal{I}$ is given by $\frac{AMC(\mathcal{I},s)}{\sum_{s' \in \mathcal{S}} AMC(\mathcal{I},s')}$ where the *absolute marginal contribution* (AMC)[9] of s is:

$$AMC(\mathcal{I}, s) = \begin{cases} \log_{10} \frac{\mathsf{PAR}_{10}(\mathcal{I}, VBS \setminus \{s\})}{\mathsf{PAR}_{10}(\mathcal{I}, VBS)} & \text{if } \mathsf{PAR}_{10}(\mathcal{I}, VBS \setminus \{s\}) > \mathsf{PAR}_{10}(\mathcal{I}, VBS) \\ 0 & \text{otherwise} \end{cases}$$

where $VBS \setminus \{s\}$ is the virtual best solver of the portfolio $\mathcal{S} \setminus \{s\}$.

If we look at the average RMC results of Tab. 2, the RF approaches dominate because they are far better on the easiest instances. We claim that measuring the marginal contribution with evaluation metrics such as the RMC or the *Shapley value* (Fréchette et al., 2016) is interesting and should be considered when *building* a meta-solver, because in this case we are interested in using a portfolio of individual solvers covering as many instances as possible.

9. Note that according to the definition given in this paper, AMC is a relative metric.

However, we must be careful when using the marginal contribution to assess the performance, especially when evaluating meta-solvers. For example, the RMC metric may lead to paradoxical results by over-penalizing approaches that work very well in general but do not excel for a few instances. Consider for example a scenario with 100 instances $\mathcal{I} = \{i_0, \ldots, i_{99}\}$ and 2 meta-solvers A and B where A solves only i_0 in 200 seconds while B cannot solve i_0, but solves $i_1, \ldots, i_{99}$ in 1 second.[10] In this setting, B clearly wins according to all the metrics so far introduced. But if we introduce a new meta-solver C performing the same as B, then A would be the new winner because the portfolios $\{A, C\}$ and $\{A, B\}$ would perform better than $\{B, C\}$.

2.3 Optimization Problems

So far we have mainly talked about evaluating meta-solvers on decision problems. While the MZNC score also considers optimization problems, for the closed gap or the speedup the generalization is not as obvious as it might seem. Here using the runtime might not be the right choice: often, a solver cannot prove the optimality of a solution, even when it actually finds it. Hence, the apparent alternative is to consider just the objective value of a solution. Nevertheless, this value needs to be *normalized*, and what bounds should we choose to do so? Furthermore: how to reward a solver that actually proves the optimality of a solution? Moreover, how to penalize solvers that cannot find any solution?

If solving to optimality is not rewarded, metrics such as the incomplete score of the MZNC or the MaxSAT Evaluation, or the satisficing track score of the 2018 International Planning Competition (*International Planning Competition*, 2018)[11] can be used. More elaborate scoring systems can combine together the quality of the best solution found, how quickly any solution is found, whether a solution is optimal, and how quickly good solutions are found. For example, Amadini, Gabbrielli, and Mauro (2016) proposed a relative metric where each solver s gets a reward in $\{0\} \cup [\alpha, \beta] \cup \{1\}$ according to the objective value $\texttt{obj}(s, i, \tau)$ of the best solution it finds, with $0 \leq \alpha \leq \beta \leq 1$. If no solution is found then s scores 0, if it solves i to optimality it scores 1, otherwise the score is computed by linearly scaling $\texttt{obj}(s, i, \tau)$ in $[\alpha, \beta]$ according to the best and worst objective values found by any other available solver on problem i. The choice of α and β determines respectively how much to penalize (reward) a solver that does not find any solution (solves to optimality). If $\alpha = 0$ and $\beta = 1$, we somehow have an incomplete score where no reward nor penalization is given. This may exacerbate small differences between solvers. Suppose indeed that s_1, s_2, s_3 find minimal values $98, 99, 100$ respectively. They would score $1, 0.5, 0$ respectively, and in particular s_3 would score 0 points, the same score as a solver not finding any solution, despite being only two units away from the best solution found by s_1. In this case, metrics like the incomplete score of the MaxSAT Evaluation would be more reasonable (the scores, in this case, would be $1, 0.99, 0.97$).

The above considerations apply to both solvers and meta-solvers. If we focus on meta-solvers only, we emphasize the importance of tracking the *sub-optimal* solutions found by the individual solvers along the search process. For example, none of the optimization scenarios

10. Example from `https://pdfs.semanticscholar.org/a0a9/403524ae9f4bc07b683c0b2ac73975511c6d.pdf`

11. This track includes optimization problems where the goal is to minimize the length of a plan.

of the ASlib report sub-optimal solutions. This is not ideal for different reasons. First of all, we cannot consider the value of metrics such as the *area* score, adopted by the MZNC since 2017, which computes the integral of a step function of the solution value over the runtime horizon. Most importantly, we cannot properly build and evaluate meta-solvers that *schedule* more than one individual solver in the solving time window $[0, \tau)$, because we do not know the best solution found by a solver at a time point $t < \tau$. This is unfortunate since scheduling different solvers is very common for a number of effective meta-solvers (Hula, Mojzísek, & Janota, 2021; Gonard, Schoenauer, & Sebag, 2017; H. Hoos, Kaminski, Lindauer, & Schaub, 2015; Amadini, Gabbrielli, & Mauro, 2014; Malitsky, Sabharwal, Samulowitz, & Sellmann, 2013; Xu, Hutter, Hoos, & Leyton-Brown, 2008).

2.4 Randomness and Aggregation

We conclude the section with some remarks about randomness and data aggregation.

When evaluating a meta-solver s on scenario $(\mathcal{I}, \mathcal{S}, \tau)$, it is common practice to partition $\mathcal{I}$ into a training set $\mathcal{I}_{tr}$, on which s "learns" how to leverage its individual solvers, and a test set $\mathcal{I}_{ts}$ where the performance of s on unforeseen problems is measured. In particular, to prevent overfitting, it is possible to use a k-fold cross validation by first splitting $\mathcal{I}$ into k disjoint folds, and then using, in turn, one fold as test set and the union of the other folds as the training set. In the 2015 AS challenge (Lindauer et al., 2019) the submissions were evaluated with a 10-fold cross validation, while in the OASC in 2017 the dataset of the scenarios was divided only into one test set and one training set. As also underlined by the organizers of the competition, this is risky because it may reward a lucky meta-solver performing well on that split but poorly on other splits.

Note that, so far, we have assumed deterministic solvers, i.e., solvers providing the same outcome if executed on the same instance in the same execution environment. Unfortunately, the scenario may contain randomized or parallel solvers, potentially producing different results with high variability. In this case, solvers should be evaluated over a number of runs, and particular care must be taken because the assumption that a solver can never outperform the VBS would be no longer valid.

A cautious choice to decrease the variance of model predictions would be to repeat the k-fold cross validation $n > 1$ times with different random splits. However, this might imply a tremendous computational effort—the training phase of a meta-solver might take hours or days—and therefore significant energy consumption. This issue is becoming an increasing concern. For example, in their recent work Matricon, Anastacio, Fijalkow, Simon, and Hoos (2021) propose an approach to early stop running an individual solver that is likely to perform worse than another solver on a subset of the instances of the scenario. In this way, fewer resources are wasted for solvers that most likely will not bring any improvement.

Finally, we spend a few words on the aggregation of the results. It is pretty common to use the arithmetic mean, or just the sum, when aggregating the outcomes of a meta-solver over different problems of the same scenario (e.g., when evaluating the results on the $n \cdot k$ test sets of a k-fold cross validation repeated n times). The same applies when evaluating different scenarios. The choice of how to aggregate the metric values into a unique value should, however, be motivated since the arithmetic mean can lead to misleading conclusions when summarizing a normalized benchmark (Fleming & Wallace, 1986). For example, to

dampen the effect of outliers, one may use the median or the geometric mean to average over normalized numbers.

3. Conclusions

The choice of different yet reasonable evaluation metrics can have opposite effects when assessing the performance of (meta-)solvers. The comparison of meta-solver approaches poses new challenges due to the diversity of the scenarios, which can remarkably vary in terms of the number and type of problems, solvers, and solving time budget.

The goal of this research note is not to propose an unrealistic *fits-all* metric but to bring attention to the problem of selecting a "stable" metric for comparing meta-solver approaches, trying to avoid under- and over-penalties as much as possible. The take-away messages and food for thought of this paper are the following:

- When selecting a metric for evaluating meta-solvers, we should first frame the target: are we approaching the problem from the developer's or the user's point of view? In the first case, where we focus on *creating* a meta-solver or even a "meta-meta-solver" (Tornede, Gehring, Tornede, Wever, & Hüllermeier, 2022), metrics like the marginal contribution can shed some lights on which (meta-)solvers work best on what instances. Here an approach performing poorly overall but much better than others over a few instances is considered valuable. However, this is not the focus of this paper.

  In this work, we put ourselves in the user's shoes by considering meta-solvers as "black boxes" for which the overall performance matters, rather than in its internals. In this context, we should ask ourselves whether the performance metric should depend on the outcome of other solvers (what we called *relative* metric) or not (*absolute* metric). The choice here is arbitrary, but one thing to consider in both cases is the *variability* of the scenarios where the meta-solvers are assessed. For example, for PAR_λ score, care must be taken because the $\tau \times \lambda$ penalty is susceptible to the timeout τ, which can vary significantly across different scenarios.

- Relative metrics are particularly exposed to the risk of amplifying small "absolute" performance variations into large differences, and it is fundamental to understand why and when this can happen. This aspect is essential for meta-solvers, for which we claim that a little time overhead should be negligible (e.g., due to feature selection or solvers' selection and scheduling). Certainly, the primary goal of meta-solvers is closing the gap between the single best solver and the virtual best solver of the portfolio of available solvers. A crucial point of this paper is to mind about how this gap is measured.

- When dealing with optimization problems it is of paramount importance to know the *sub-optimal* solutions found by individual solvers along the search. Indeed, many effective meta-solvers rely on scheduling more than one solver for a limited time, and without knowing the anytime performance of a solver it is impossible to assess the performance of the schedule. Unfortunately, standard benchmarks like ASlib lack this information. We believe that a community effort is needed to fill this gap.

- Meta-solvers are typically built by learning on a training set and testing the learned model on a disjoint test set. It is essential to use approaches like cross-validation to increase the stability of the results, possibly using a *randomly repeated* cross-validation to avoid rewarding "lucky" approaches overfitting on a particular training/testing set. Also, the problem of aggregating the results over different runs should not be overlooked: we naturally tend to consider the sum or the arithmetic mean to determine the overall performance, but other choices may make (more) sense.

References

Amadini, R., Gabbrielli, M., & Mauro, J. (2014). SUNNY: a Lazy Portfolio Approach for Constraint Solving. *TPLP*, *14*(4-5), 509–524.

Amadini, R., Gabbrielli, M., & Mauro, J. (2015). SUNNY-CP: a sequential CP portfolio solver. In *SAC* (pp. 1861–1867). ACM.

Amadini, R., Gabbrielli, M., & Mauro, J. (2016). Portfolio approaches for constraint optimization problems. *Annals of Mathematics and Artificial Intelligence*, *76*(1-2), 229–246.

Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, M., Malitsky, Y., Fréchette, A., ... Vanschoren, J. (2016). Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, *237*, 41–58.

Fleming, P. J., & Wallace, J. J. (1986). How not to lie with statistics: The correct way to summarize benchmark results. *Commun. ACM*, *29*(3), 218–221. Retrieved from https://doi.org/10.1145/5666.5673 doi: 10.1145/5666.5673

Fréchette, A., Kotthoff, L., Michalak, T. P., Rahwan, T., Hoos, H. H., & Leyton-Brown, K. (2016). Using the shapley value to analyze algorithm portfolios. In D. Schuurmans & M. P. Wellman (Eds.), *Proceedings of the thirtieth AAAI conference on artificial intelligence, february 12-17, 2016, phoenix, arizona, USA* (pp. 3397–3403). AAAI Press.

Gonard, F., Schoenauer, M., & Sebag, M. (2017, 11–12 Sep). Asap.v2 and asap.v3: Sequential optimization of an algorithm selector and a scheduler. In M. Lindauer, J. N. van Rijn, & L. Kotthoff (Eds.), *Proceedings of the open algorithm selection challenge* (Vol. 79, pp. 8–11). Brussels, Belgium: PMLR.

Hoos, H., Kaminski, R., Lindauer, M., & Schaub, T. (2015). aspeed: Solver scheduling via answer set programming 1. *Theory and Practice of Logic Programming*, *15*(1), 117–142.

Hoos, H. H. (2012). Automated algorithm configuration and parameter tuning. In Y. Hamadi, É. Monfroy, & F. Saubion (Eds.), *Autonomous search* (pp. 37–71). Springer.

Hula, J., Mojzísek, D., & Janota, M. (2021). Graph neural networks for scheduling of SMT solvers. In *33rd IEEE international conference on tools with artificial intelligence, ICTAI 2021, washington, dc, usa, november 1-3, 2021* (pp. 447–451). IEEE. Retrieved from https://doi.org/10.1109/ICTAI52525.2021.00072 doi: 10.1109/ICTAI52525.2021.00072

ICAPS. (2021). *The international planning competition web page.* https://www.icaps-conference.org/competitions/. (Accessed: 2021-12-10)

International Planning Competition. (2018). (Available at https://ipc2018-classical.bitbucket.io/)

Kotthoff, L. (2016). Algorithm selection for combinatorial search problems: A survey. In *Data mining and constraint programming* (pp. 149–190). Springer.

Kotthoff, L., Hurley, B., & O'Sullivan, B. (2017). The ICON challenge on algorithm selection. *AI Magazine*, *38*(2), 91–93.

Lindauer, M., van Rijn, J. N., & Kotthoff, L. (2019). The algorithm selection competitions 2015 and 2017. *Artificial Intelligence*, *272*, 86–100.

Liu, T., Amadini, R., Mauro, J., & Gabbrielli, M. (2021). sunny-as2: Enhancing SUNNY for algorithm selection. *J. Artif. Intell. Res.*, *72*, 329–376.

Luo, C., Vallati, M., & Hoos, H. H. (2018). *Sparkle sat challenge 2018.* https://ada.liacs.nl/events/sparkle-sat-18/. (Accessed: 2023-01-05)

Luo, C., Vallati, M., & Hoos, H. H. (2019). *Sparkle planning challenge 2019.* https://ada.liacs.nl/events/sparkle-planning-19/. (Accessed: 2023-01-05)

Malitsky, Y., Sabharwal, A., Samulowitz, H., & Sellmann, M. (2013). Boosting sequential solver portfolios: Knowledge sharing and accuracy prediction. In G. Nicosia & P. M. Pardalos (Eds.), *Learning and intelligent optimization - 7th international conference, LION 7, catania, italy, january 7-11, 2013, revised selected papers* (Vol. 7997, pp. 153–167). Springer. Retrieved from https://doi.org/10.1007/978-3-642-44973-4_17 doi: 10.1007/978-3-642-44973-4_17

Matricon, T., Anastacio, M., Fijalkow, N., Simon, L., & Hoos, H. H. (2021). Statistical comparison of algorithm performance through instance selection. In L. D. Michel (Ed.), *27th international conference on principles and practice of constraint programming, CP 2021, montpellier, france (virtual conference), october 25-29, 2021* (Vol. 210, pp. 43:1–43:21). Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

MaxSAT Evaluation. (2022). (Available at https://maxsat-evaluations.github.io/2022/index.html)

QBFEVAL. (2021). *Qbf evaluations web page.* http://www.qbflib.org/index_eval.php. (Accessed: 2021-12-10)

SAT competition. (2021). *The international sat competition web page.* http://www.satcompetition.org/. (Accessed: 2021-12-10)

Stuckey, P. J., Becket, R., & Fischer, J. (2010). Philosophy of the minizinc challenge. *Constraints An Int. J.*, *15*(3), 307–316. Retrieved from https://doi.org/10.1007/s10601-010-9093-0 doi: 10.1007/s10601-010-9093-0

Stuckey, P. J., Feydy, T., Schutt, A., Tack, G., & Fischer, J. (2014). The MiniZinc Challenge 2008-2013. *AI Magazine*, *35*(2), 55–60. Retrieved from http://www.aaai.org/ojs/index.php/aimagazine/article/view/2539

Tornede, A., Gehring, L., Tornede, T., Wever, M., & Hüllermeier, E. (2022). Algorithm selection on a meta level. *Machine Learning*, 1–34.

XCSP Competition. (2019). *Xcsp competition.* http://www.cril.univ-artois.fr/XCSP19/. (Accessed: 2021-12-10)

Xu, L., Hutter, F., Hoos, H., & Leyton-Brown, K. (2008). Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, *32*, 565–606.