EasyChair Preprint

№ 8627

A Portfolio-Based Analysis Method for Competition Results

Nguyen Dang

August 9, 2022

A portfolio-based analysis method for competition results

Nguyen Dang

University of St Andrews, United Kingdom

Abstract

Competitions such as the MiniZinc Challenges or the SAT competitions have been very useful sources for comparing performance of different solving approaches and for advancing the state-of-the-arts of the fields. Traditional competition setting often focuses on producing a ranking between solvers based on their average performance across a wide range of benchmark problems and instances. While this is a sensible way to assess the relative performance of solvers, such ranking does not necessarily reflect the full potential of a solver, especially when we want to utilise a portfolio of solvers instead of a single one for solving a new problem. In this paper, I will describe a portfolio-based analysis method which can give complementary insights into the performance of participating solvers in a competition. The method is demonstrated on the results of the MiniZinc Challenges and new insights gained from the portfolio viewpoint are presented.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases algorithm portfolio, algorithm selection, constraint programming

Funding *Nguyen Dang*: is a Leverhulme Early Career Fellow

1 Introduction

In many algorithmic communities, competitions have been the sources for comparing both established and newly developed solving techniques and for advancing the state-of-the art solving approaches over the years. Examples include the SAT competitions [11, 4] for SAT solvers, the MiniZinc Challenge [19, 20] for constraint solving approaches, the International Planning competitions for the planning community [14, 21]. In a typical competition setting, each participating solver is evaluated on a set of benchmark problems and its average performance across the whole benchmark set is recorded. A ranking among all participants is then produced based on the collected average performance of each solver. While this is a sensible way to assess the relative performance of participating solvers, such ranking does not necessarily reflect the full potential of a solver. For example, a solver with a low overall rank but uniquely performs well on a small subset of the benchmark problems may still be very useful in practice. This is particularly true in the context of algorithm portfolios and algorithm selection [13, 12], where the aim is construct and utilise a set (a portfolio) of solvers with complementary strengths.

There are recent initiatives [10] that call for a new type of "cooperative" competitions such as the Sparkle SAT challenge 2018 [15] and the Sparkle Planning challenge 2019 [16]. In those challenges, an algorithm selector is built on a portfolio of all participating solvers. Each solver is then ranked based on its *marginal contribution* [22] to the performance of the selector. More specifically, given a portfolio of n algorithms $\mathcal{A} = \{a_1, a_2, .., a_n\}$, the marginal contribution of an algorithm a_k to $\mathcal{A}$ on an instance set $\mathcal{I}$ is defined as $P(\mathcal{A}, \mathcal{I}) - P(\mathcal{A} \setminus \{a_k\}, \mathcal{I})$, where $P(\mathcal{A}, \mathcal{I})$ is the performance of an algorithm selector constructed from $\mathcal{A}$ measured on $\mathcal{I}$. Such type of competitions gives a different view on the performance of a solving approach and can give credits to the solvers that appear to be weak in a traditional competition setting. The algorithm selector will act as the new state-of-art (SOTA) by standing on the shoulders of the participating solvers, and the aim of a newly developed solver would be to join the

SOTA portfolio and improve the SOTA performance via its marginal contribution to the current portfolio.

The Sparkle setting brings a new and interesting view to competition organisation. However, there are two things we need to consider when applying the setting to a new domain. Firstly, the usage of an algorithm selector for measuring contribution of each solver assumes that we have a pre-defined set of instance features across all possible problems of the domain, and that those features are predictive of all participating solvers' performance. Those assumptions have been long shown to hold for SAT instance features such as the ones used in the SATZilla system [23]. However, such assumptions may not always be met when considering a new problem domain, or even on new solvers within the current domain. Secondly, there has been criticism on the marginal contribution measurement used by the challenges. Given a portfolio $\mathcal{A}$, imagine the case where we have two solvers a_1 and a_2 in $\mathcal{A}$, both of which perform equally better than each of the rest of $\mathcal{A}$ on the same subset of instances of $\mathcal{I}$. If the size of such subset is large, it means that the average performance of a_1 and a_2 are quite strong compared to the others. However, the marginal contribution of both solvers to $\mathcal{A}$ will be zero, which does not give them the true credits for their overall performance.

Another method for measuring the importance of a solver in a portfolio based on the *Shapley values* [18], a concept taken from coalitional game theory, was proposed in [7]. This measure takes into account the marginal contribution of each solver to *every subset of the portfolio* rather than just the full portfolio. The Shapley value of a solver is the average value of those contributions. In [7], the authors calculated the Shapley values of participating solvers on several SAT competitions, and show that the resulting ranking can be quite different from the official ranks of the competitions, which reveal additional interesting insights into the competition results.

Following those ideas, in this paper, I will present a portfolio-based method for analysing competition results. In contrast to the Sparkle setting, I will consider a simplified algorithm portfolio setting where we make use of a portfolio by just running all solvers in parallel on each given instance, and we stop when one of the solvers has solved the instance. This setting is extremely easy to use, and is not too impractical due to the popularity of multiple-core processors and high performance computing systems nowadays. Moreover, in all previous works, only running time is used as solver performance (runs where instances are unsolved are penalised using PAR2 or PAR10, which is equivalent to multiplying the timeout limit by a factor of 2 or 10), while the analysis in this paper will take into account both running time and solution quality as performance measure by utilising the MiniZinc challenge scoring method [1].

More concretely, the analysis consists of three steps:

- Step 1: find the smallest portfolios that can achieve the best possible performance. This step will tell us whether we need all solvers to achieve the best performance, or if there is a small subset of solvers that completely dominate the rest (Section 3).
- Step 2: given the portfolios found in step 1, find the best subset of solvers for each subset size. This will give us an idea about the trade-off between reducing the number of solvers in a portfolio and the resulting performance (Section 4).
- Step 3: measure the importance of each solver under the portfolio viewpoint using the Shapley values. This step provides a summary of the observations in step 2 (Section 5).

[1] `https://www.minizinc.org/challenge2021/rules2021.html`

I will analyse the results of the MiniZinc Challenges based on the steps described above and show the new insights gained from such portfolio viewpoint compared to the traditional ranking method. However, I am not proposing to replace the current ranking methods of the MiniZinc Challenges with portfolio-based rankings. This analysis is rather a complementary viewpoint to the current results of the challenges and may be useful for the algorithm developers and the community when adopting those solvers for their specific use cases. The source code and data used in the analysis are available on github at `https://github.com/ndangtt/portfolio-based-analysis`.

2 Background

This section describes the concepts and terminologies that will be used throughout the whole analysis, including: (i) the scoring method used by the MiniZinc Challenges(Section 2.1); (ii) the definition of the Oracle and the Participant-Oracle (Section 2.2), the two baselines used for measuring performance of a portfolio of solvers from a competition; and (iii) the performance measure of a solver portfolio (Section 2.3).

2.1 MiniZinc Scoring Method

The Sparkle challenges and many algorithm selection systems typically focuses on decision problems, where the performance measure only takes into account the running time of each solver (with penalty on timeout runs). However, for optimisation problems, solution quality is another important aspect of performance. The MiniZinc scoring method is a nice way to aggregate both running time and solution quality into a single measure. Given two solvers A and B, a problem constraint model P and an instance I of the same problem, the MiniZinc scoring method assigns a score to each of the two solvers such that the better performing one gets a higher score, and the sum of the two scores is equal to 1. More concretely, if P is a decision problem, solver A is considered better than solver B on an instance I if: (i) A can solve I within the time limit while B cannot (in such case A get the full score of 1 and B gets 0 score); or (ii) I is solvable by both solvers within the time limit and A is faster than B (in such case the scores of each solver is proportional to the other solver's running time). If P is an optimisation problem, A is better than B if: (i) A can solve I to optimality while B cannot, or (ii) A produces better final solution quality than B; or (iii) both solvers produce the same final solution quality or both solve the instance to optimality but A is faster than B. In the first two cases, A gets the full scores of 1, while in the last case, the scores of A and B are proportional to each other's running time.

The official ranking of the MiniZinc Challenges is based on the scoring method described above and the Borda counting system [6]. For each instance, the method is applied to every pair of solvers. The final score of each solver is its average score across all instances of the competition and solvers with higher scores get better ranks. Note that in the MiniZinc scoring method, if both solvers fail to solve an instance, the first one of the pair will get a score of 1 while the second one gets 0. This design is on purpose due to the fact that the Borda counting will calculate the scores of the same pair in both directions. Eventually both solvers will get the same total score of 1, indicating that their performance is indistinguishable on the given instance.

2.2 The Oracle and the Participant-Oracle

Given an algorithm portfolio $\mathcal{A}$, the Virtual Best Solver of $\mathcal{A}$, denoted as $\mathcal{VBS}(\mathcal{A})$ is defined as the hypothetical best solver that we can obtain from the portfolio. This can be achieved by either having a perfect selector that can choose the best performing algorithm for a given instance, or by simply running all algorithms in the portfolio in parallel.

In the MiniZinc Challenges, there are a number of solvers that are not participating in the competition, but their performance on the all benchmark instances are used together with the participating solvers' performance data when calculating the Borda scores for the competition ranking. In our analysis, we will consider two scenarios in a competition, one where only participating solvers are considered, and one where non-participating solvers are also included. The VBS of the first one will be called the Oracle, denoted as $\mathcal{O}$, while the VBS of the second one, namely the Participant-Oracle is denoted as $\mathcal{O}_{par}$. Obviously, performance of the former one is at least as good as the latter one. Those two oracles will be used as the baselines for the analysis in this paper. More concretely, they are for measuring how good a portfolio is, as detailed in the next part of this section.

2.3 Measuring a Portfolio's Performance

Given a pair of portfolios $\mathcal{A}_1$ and $\mathcal{A}_2$, we can compare the performance of the two portfolios by calculating the total MiniZinc scores of $\mathcal{VBS}(\mathcal{A}_1)$ and $\mathcal{VBS}(\mathcal{A}_2)$ across all instances. The ratio between the two scores will tell us how much one portfolio is better than another.

In a competition setting, the performance of an arbitrary portfolio $\mathcal{A}$ w.r.t. the Oracle $\mathcal{O}$ can be defined as:

$$\mathcal{P}_{\mathcal{O}}(\mathcal{A}) = score(\mathcal{VBS}(\mathcal{A}))/score(\mathcal{O}) \tag{1}$$

where $score(.)$ is the total MiniZinc scores calculated for the pair of $\mathcal{A}$ and $\mathcal{O}$ across all competition instances. Note that $\mathcal{P}_{\mathcal{O}}(\mathcal{A}) \leq 1$ since $\mathcal{A} \subseteq \mathcal{O}$. If $\mathcal{A}$ only consists of the participating solvers, we can also measure the performance of $\mathcal{A}$ w.r.t. the Participant-Oracle $\mathcal{O}_{par}$:

$$\mathcal{P}_{\mathcal{O}_{par}}(\mathcal{A}) = score(\mathcal{VBS}(\mathcal{A}))/score(\mathcal{O}_{par}) \tag{2}$$

We know that $\mathcal{O}_{par}$ is never better than $\mathcal{O}$. Using the performance measure described in this part, the performance difference between the two can be quantified. Figure 1 shows $\mathcal{P}_{\mathcal{O}}(\mathcal{O}_{par})$ (per track) for the nine MiniZinc Challenges from 2013 to 2021. We can see that the participating solvers never dominate the non-participating ones on the competition instance set. In fact, in most cases, $\mathcal{O}_{par}$ achieves well below 80% the performance of $\mathcal{O}$. The ratio is particular low for the year 2016 ($< 40\%$ for all tracks) and for the FD track of the year 2020 (13.5%), indicating significant rooms for improvement in the performance of the participating solvers on the competition benchmark instance sets.

3 Minimum-sized Portfolios with Oracle Performance

In the first step of the analysis, we will look into finding the smallest portfolio(s) that can achieve the Oracle performance. This is equivalent to solving the set cover problem. More specifically, given a portfolio of n algorithms $\mathcal{A} = \{a_1, a_2, ..., a_n$ and a set of instances $\mathcal{I}$, we can define $\mathcal{I}_k \subseteq \mathcal{I}$ as the set of instances on each of which algorithm a_k is the best performing solver. We want to find a portfolio $\mathcal{A}^* \subseteq \mathcal{A}$ such that $\bigcup_{a_k \in \mathcal{A}^*} \mathcal{I}_k = \mathcal{I}$ and $|\mathcal{A}^*|$ is minimised.

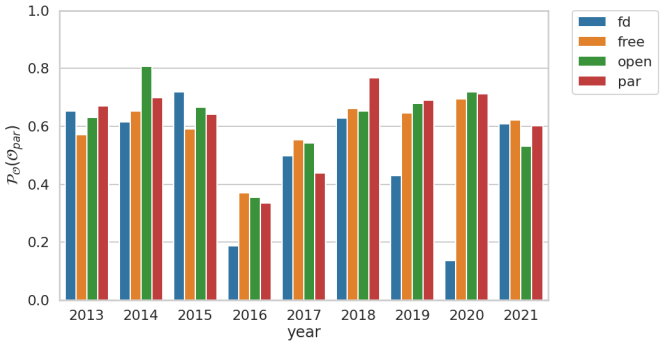

Figure 1 Performance of the Participant-Oracle w.r.t. the Oracle for the MiniZinc Challenges 2013–2021. Results are for three tracks of the competition: FD, FREE and PAR.

In this section, the set cover problem is written in the constraint modelling language ESSENCE [8] and is solved via the ESSENCE Pipeline [1], whose the solving procedure includes a translation of model and instance into solver input format using the automated modelling tools CONJURE [2] and SAVILEROW [17] followed by a call to the constraint solver MINION [9]. This problem is solved for all four tracks of the MiniZinc competitions 2013–2021. We will look into two scenarios, one where only participant solvers are considered, and one where non-participants are also included. It only takes a few seconds for the ESSENCE pipeline to solve each set cover problem instance in our case. Interestingly, for every case, only one single optimal solution is found, i.e., the minimum-sized portfolio with Oracle performance is unique.

First, we will look at the scenario with participant solvers only. Figure 2 shows the ratio of the minimum-sized found to the number of all solvers. The ratios are consistently high, with many cases well above 80%. Notably, they are equal to 100% for the FD track of 6/9 years. Those numbers suggest that the participant solvers are often complementary to each other, and they should be used together if possible to achieve the best possible performance.

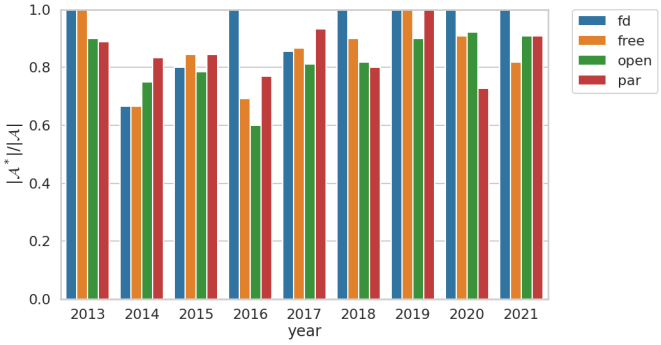

Figure 2 $|\mathcal{A}^*|/|\mathcal{A}|$, where $\mathcal{A}$ is the set of all participant solvers in the competitions, and $\mathcal{A}^*$ is the minimum-sized portfolio with Participant-Oracle performance (i.e., $\mathcal{P}_{\mathcal{VBS}(\mathcal{A})}(\mathcal{A}^*) = 1$)

Now we will look into the case where non-participants are also included in the portfolio. The percentages of the minimum numbers of solvers needed to achieve the Oracle performance are shown in Figure 4. In contrast to the previous scenario, the ratios are now mostly below 60%, with a few cases where an amount of less than 40% of solvers is sufficient to cover the

whole portfolio's performance. This observation suggests that some solvers are completely dominated by others and can be removed from the original portfolio without affecting its overall performance. A closer look into the proportions of participants and non-participants in the minimum-sized portfolios found indicate complementary strengths between both solver groups. In fact, for most cases the non-participant solvers never completely dominate the participant ones as shown by the presence of the blue color in all charts of Figure 4, with the two exceptions of track FREE and track OPEN of year 2016.

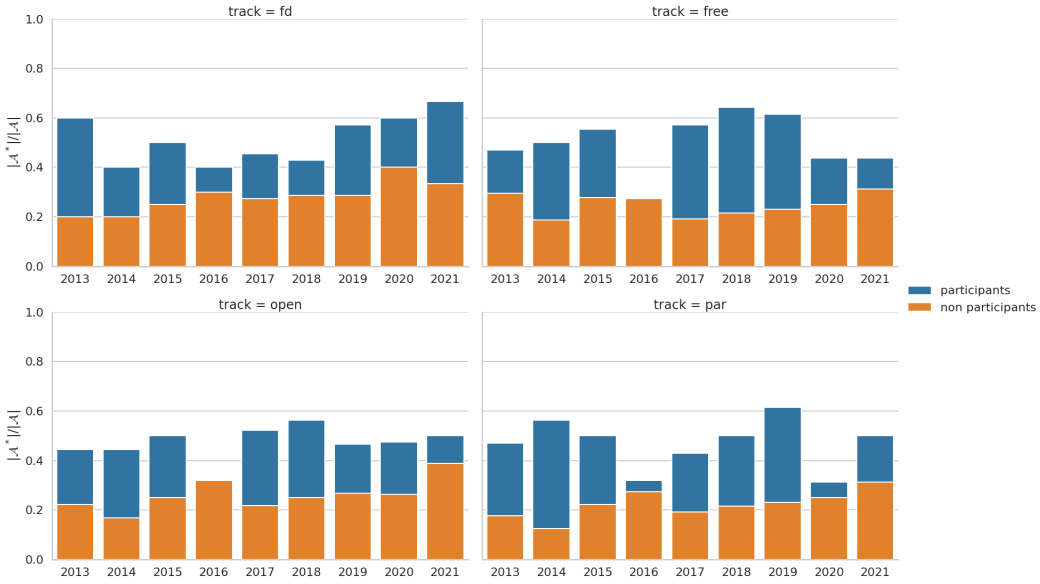

Figure 3 $|\mathcal{A}^*|/|\mathcal{A}|$, where $\mathcal{A}$ is the set of all solvers including the non-participants, and $\mathcal{A}^*$ is the minimum-sized portfolio with Oracle performance. Each bar also shows the amount of participants and non-participants being chosen in $\mathcal{A}^*$.

4 Trade-off between portfolio size and performance

The analysis in the previous section tells us that many solvers have complementary strengths and we need a good number of them being included in the portfolio to achieve the best possible performance. Even in the second scenario (where the non-participants are included), although the percentages shown in the plots are less than 40%, the actual numbers of required solvers range from 5 to 17 due to the large size of the original portfolio. What happens if we can only afford a handful of solvers due to resource limit? The second step of the analysis, presented in this section, will investigate the trade-off between reducing the number of solvers and its impact on the resulting portfolio's performance. From now on, for brevity, I will focus on one track of the MINIZINC Challenges, the FREE track, but the same analysis can be applied to any other tracks.

Given the minimum-sized portfolio with Oracle performance $\mathcal{A}^*$ found in the previous step and a portfolio size $k \leq |\mathcal{A}^*|$, we can use brute-force search to find the best subset of solvers with size k. More concretely, we find $\mathcal{K}^* \subseteq \mathcal{A}^*$ such that $\mathcal{K}^* = \operatorname{argmax}_{\mathcal{K} \subseteq \mathcal{A}^*, |\mathcal{K}|=k} P_{\mathcal{VBS}(\mathcal{A}^*)}(\mathcal{K})$. Figure 4 shows the proportions of solvers needed to achieve various levels of the Oracle performance for the FREE track of all years 2013–2021. In most cases, we need around 50% of the solvers in $\mathcal{A}^*$ to achieve at least 80% performance of the whole portfolio. And in

almost all cases, an amount of less than 80% of $|\mathcal{A}^*|$ is sufficient to achieve at least 95% of the Oracle performance.

Figure 4 The minimum proportions of solvers in $\mathcal{A}^*$ needed to achieve 80%, 90%, and 95% Oracle performance (track FREE). Top: participants only, bottom: non-participants also included.

A closer look into the best subset of solvers for each portfolio size k reveals some interesting facts. Table 1 lists the best participant subsets of years 2019–2021 for each size k and their performance wrt the Participant-Oracle. OR-Tools appears in every single subset, indicating its superior performance on the competition datasets, in other words, if we can only pick a few solvers to use in a portfolio, OR-Tools should definitely be included. This is also inline with the competition rankings as OR-Tools got the gold medals for the FREE track of all three years. Interestingly, for year 2019, the competition's bronze-medal solver Picat-SAT is suggested to be included alongside with OR-Tools for $k = 2$ rather than the silver-medal solver SICStus Prolog. Results of year 2020 suggests that flatzingo has very good complementary power to the winner OR-Tools, as it was included in all portfolio sizes from 2 onward. Notably, a combination of only two solvers OR-Tools and flatzingo can already reach 70% the performance of the full set of all participants. This is despite the fact that flatzingo was only ranked 4^{th} in the competition. A similar observation is seen for the solver Yuck in the 2021's results, using Yuck alongside with OR-Tools can help to boost the performance to 12%, although Yuck was ranked second to last in the competition, and is the last one within the minimum-sized portfolio with Oracle performance.

5 Portfolio-based solver importance with Shapley values

In the previous section, we have looked into the trade-off between portfolio size and the resulting performance, which would help us to choose the best portfolio when we can only

$\mathcal{P}$	$\mathcal{K}^*$
	year: 2019, track: free
36.1%	or-tools,
55.6%	or-tools, picatsat
67.4%	or-tools, picatsat, sicstus
79.2%	or-tools, picatsat, yuck, sicstus
91.5%	or-tools, picatsat, izplus, yuck, sicstus
96.2%	or-tools, picatsat, izplus, yuck, jacop, sicstus
98.2%	or-tools, picatsat, izplus, yuck, concrete, jacop, sicstus
99.5%	or-tools, picatsat, izplus, yuck, concrete, oscarcbls, jacop, sicstus
100%	or-tools, picatsat, izplus, yuck, concrete, oscarcbls, jacop, choco, sicstus
	year: 2020, track: free
59.7%	or-tools,
71.7%	or-tools, flatzingo
81.0%	or-tools, sicstus, flatzingo
90.2%	or-tools, sicstus, mistral, flatzingo
94.0%	or-tools, sicstus, mistral, flatzingo, oscarcbls
96.9%	or-tools, sicstus, mistral, picatsat, flatzingo, oscarcbls
98.2%	or-tools, sicstus, mistral, picatsat, choco, flatzingo, oscarcbls
99.4%	or-tools, jacop, sicstus, mistral, picatsat, choco, flatzingo, oscarcbls
99.8%	or-tools, jacop, sicstus, mistral, picatsat, choco, flatzingo, optimathsat-int, oscarcbls
100%	or-tools, jacop, sicstus, mistral, picatsat, choco, flatzingo, optimathsat-int, oscarcbls, yuck
	year: 2021, track: free
49.8%	or-tools-cp-sat,
62.0%	or-tools-cp-sat, yuck
75.6%	or-tools-cp-sat, picatsat, yuck
82.9%	or-tools-cp-sat, picatsat, choco-4-10-7, yuck
88.5%	or-tools-cp-sat, picatsat, choco-4-10-7, jacop, yuck
92.1%	or-tools-cp-sat, picatsat, coin-or-cbc, choco-4-10-7, jacop, yuck
95.6%	izplus, or-tools-cp-sat, picatsat, coin-or-cbc, choco-4-10-7, jacop, yuck
97.6%	izplus, or-tools-cp-sat, picatsat, coin-or-cbc, mistral-2.0, choco-4-10-7, jacop, yuck
100%	izplus, or-tools-cp-sat, flatzingo, picatsat, coin-or-cbc, mistral-2.0, choco-4-10-7, jacop, yuck

Table 1 The best subset of solvers for each portfolio size k (column $\mathcal{K}^*$) and their performance vs the Participant-Oracle, i.e., $\mathcal{P}_{\mathcal{VBS}(\mathcal{A}^*)}(\mathcal{VBS}(\mathcal{K}^*))$ (column $\mathcal{P}$)

afford a limited number of solvers due to resource constraint. We have also seen that there are cases where a solver looks rather weak based on the competition ranking system is actually very well complementary to the winner and seems to have strong impact on portfolio performance. In this section, we will use the Shapley value as a summary measure for quantifying the contribution of a solver to a portfolio's performance. The idea of using the Shapley value for analysing the importance of a solver in a portfolio were proposed by the authors of [7] and were demonstrated on several SAT competitions. In this third step of the analysis, we can make use of this idea in combination with the performance measure defined in Section 2.3 for measuring the quality of each solver from a portfolio point of view, where both solution quality and running time are taken into account.

Given a portfolio $\mathcal{A}$ and a performance measure $\mathcal{P}$, the Shapley value of each solver $a \in \mathcal{A}$, denoted as $\mathcal{SL}_{\mathcal{A}}(a)$, is defined as the marginal contribution of the solver to the performance of all subsets of $\mathcal{A}$:

$$\mathcal{SL}_{\mathcal{A}}(a) = \sum_{\mathcal{K} \subseteq \mathcal{A} \setminus \{a\}} (\mathcal{P}(\mathcal{K} \cup a) - \mathcal{P}(\mathcal{K})) \tag{3}$$

Figure 5 shows the calculated Shapley values for all solvers in the minimum-sized portfolio with Participant-Oracle performance found in step 1 using $\mathcal{P}_{\mathcal{O}_{par}}(\cdot)$ as the performance measure (years 2019–2021, track FREE). The MINIZINC Borda scores of those solvers are also included in the figure for comparison. Since the Borda scores may change when solvers are removed from a portfolio, two sets of MINIZINC scores are shown in the figure: one when all participants are considered in the score calculation, and one where only the solvers in

the minimum-sized portfolio with Participant-Oracle performance are taken into account. Nevertheless, the MiniZinc rankings of solvers in $\mathcal{A}^*$ for both cases are exactly the same.

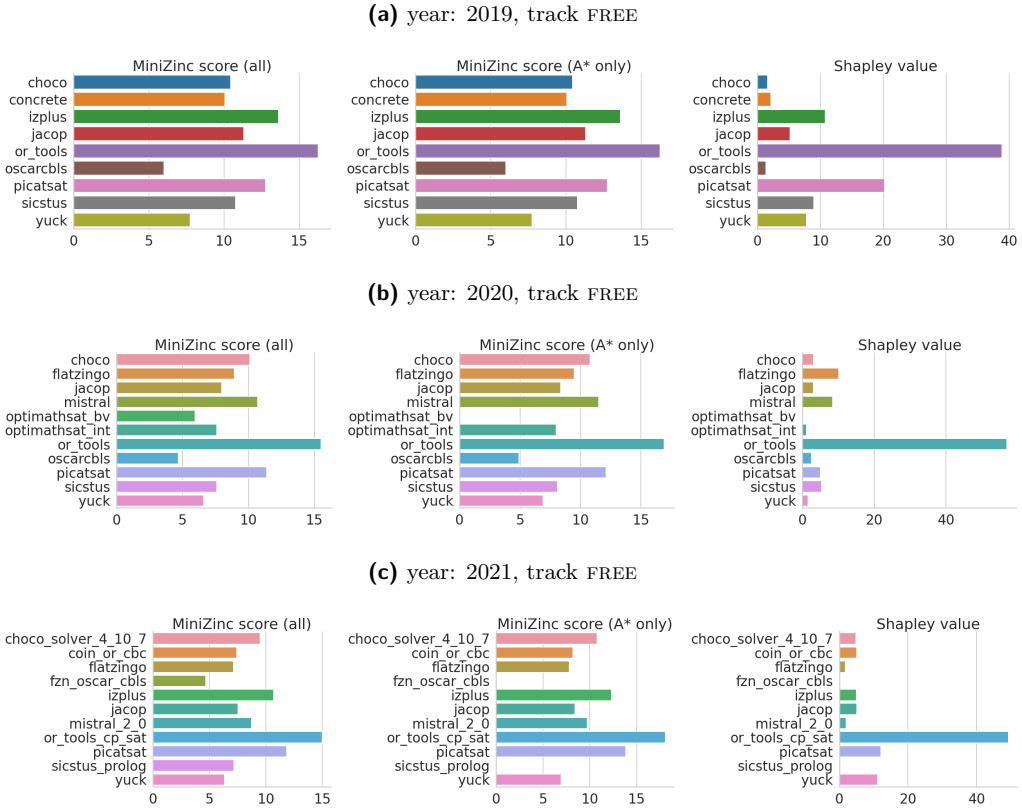

Figure 5 For each participant solver in the minimum-sized portfolio with Participant-Oracle performance ($\mathcal{A}^*$, found in step 1 of the analysis): (i) MiniZinc score (left: with all participants included, middle: with solvers in $\mathcal{A}^*$ only); and (ii) Shapley value (right)

The Shapley values confirm the significant importance of OR-Tools to the portfolio performance as observed in the previous analysis step, as both MiniZinc scores and Shapley value are the highest compared to the rest of the portfolio. Another solver with consistent high rank both in term of competition scores and Shapley value is Picat-SAT in year 2019 and 2021, although in year 2020 its rank was swapped with flatzingo when Shapley value is used. Lastly, the Shapley value of Yuck in 2021 confirms its important contribution in the portfolio setting, despite its low rank in the competition ranking system.

6 Conclusion

Traditional ranking method in competition settings is a good way to measure performance of solvers but it does not necessarily reveal the full potential of a solver. Following the ideas of the Sparkle challenges [15, 16] and the work of [7], in this paper, a three-step portfolio-based analysis method for studying solver performance in competitions is presented. The analysis makes use of the Virtual Best Solver performance and assumes the simplest setting of utilising an algorithm portfolio where all algorithms are run in parallel, hence does not require instance features or a machine learning model for predicting solver performance. A demonstration on

the MiniZinc competition results shows additional insights into the relative performance of solvers and how to choose among them given limited computational resources. The analysis provides a useful complementary viewpoint to the current competition assessment system and reveal interesting insights that were not shown in a traditional ranking system. For future work, an integration of other scoring methods besides the MiniZinc Borda counting system, such as the ones suggested in [5] and [3] can be added.

References

1 ESSENCE modelling pipeline:. https://constraintmodelling.org/.

2 Ozgur Akgun, Ian Miguel, Chris Jefferson, Alan M Frisch, and Brahim Hnich. Extensible automated constraint modelling. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

3 Roberto Amadini, Maurizio Gabbrielli, and Jacopo Mauro. Portfolio approaches for constraint optimization problems. *Annals of Mathematics and Artificial Intelligence*, 76(1):229–246, 2016.

4 Tomás Balyo, Marijn Heule, and Matti Jarvisalo. SAT competition 2016: Recent developments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

5 Frédéric Boussemart, Christophe Lecoutre, Arnaud Malapert, and Cédric Piette. About benchmarking and competitions of solvers in constraint programming. *The International Planning Competition (WIPC-15)*, page 1, 2015.

6 Yann Chevaleyre, Ulle Endriss, Jérôme Lang, and Nicolas Maudet. A short introduction to computational social choice. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 51–69. Springer, 2007.

7 Alexandre Fréchette, Lars Kotthoff, Tomasz Michalak, Talal Rahwan, Holger Hoos, and Kevin Leyton-Brown. Using the shapley value to analyze algorithm portfolios. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

8 Alan M Frisch, Matthew Grum, Christopher Jefferson, Bernadette Martínez Hernández, and Ian Miguel. The design of essence: A constraint language for specifying combinatorial problems. In *IJCAI*, volume 7, pages 80–87, 2007.

9 Ian P. Gent, Christopher Jefferson, and Ian Miguel. Minion: A fast scalable constraint solver. In *Proceedings ECAI 2006*, pages 98–102, 2006.

10 Holger H Hoos. Sparkle: A PbO-based Multi-agent Problem-solving Platform. 2015.

11 Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon. The international SAT solver competitions. *Ai Magazine*, 33(1):89–92, 2012.

12 Pascal Kerschke, Holger H Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45, 2019.

13 Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, Jim McFadden, Yoav Shoham, et al. A portfolio approach to algorithm selection. In *IJCAI*, volume 3, pages 1542–1543, 2003.

14 Derek Long and Maria Fox. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, 20:1–59, 2003.

15 Chuan Luo and Holger Hoos. Sparkle SAT Challenge, 2018. Available from https://ada.liacs.nl/events/sparkle-sat-18/.

16 Chuan Luo, Mauro Vallati, and Holger Hoos. Sparkle Planning Challenge, 2098. Available from https://ada.liacs.nl/events/sparkle-planning-19/.

17 Peter Nightingale, Özgür Akgün, Ian P Gent, Christopher Jefferson, Ian Miguel, and Patrick Spracklen. Automatically improving constraint models in Savile Row. *Artificial Intelligence*, 251:35–61, 2017.

18 L. S. Shapley. *A Value for n-Person Games*, pages 307–318. Princeton University Press, 2016.

19 Peter J Stuckey, Ralph Becket, and Julien Fischer. Philosophy of the MiniZinc challenge. *Constraints*, 15(3):307–316, 2010.

20 Peter J Stuckey, Thibaut Feydy, Andreas Schutt, Guido Tack, and Julien Fischer. The minizinc challenge 2008–2013. *AI Magazine*, 35(2):55–60, 2014.

21 Mauro Vallati, Lukas Chrpa, Marek Grześ, Thomas Leo McCluskey, Mark Roberts, Scott Sanner, et al. The 2014 international planning competition: Progress and trends. *Ai Magazine*, 36(3):90–98, 2015.

22 Lin Xu, Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. Evaluating component solver contributions to portfolio-based algorithm selectors. In *International conference on theory and applications of satisfiability testing*, pages 228–241. Springer, 2012.

23 Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *Journal of artificial intelligence research*, 32:565–606, 2008.