Using the Shapley Value to Analyze Algorithm Portfolios

Alexandre Fréchette
Dept. of Computer Science
Univ. of British Columbia, Canada
afrechet@cs.ubc.ca

Lars Kotthoff
Dept. of Computer Science
Univ.y of British Columbia, Canada
larsko@cs.ubc.ca

Tomasz Michalak
Univ. of Oxford, UK
Univ. of Warsaw, Poland
tomasz.michalak@cs.ox.ac.uk

Talal Rahwan
Masdar Inst. of Science and Technology
Abu Dhabi, United Arab Emirates
trahwan@gmail.com

Holger H. Hoos
Dept. of Computer Science
Univ. of British Columbia, Canada
hoos@cs.ubc.ca

Kevin Leyton-Brown
Dept.of Computer Science
Univ. of British Columbia, Canada
kevinlb@cs.ubc.ca

Abstract

Algorithms for NP-complete problems often have different strengths and weaknesses, and thus algorithm portfolios often outperform individual algorithms. It is surprisingly difficult to quantify a component algorithm's contribution to such a portfolio. Reporting a component's standalone performance wrongly rewards near-clones while penalizing algorithms that have small but distinct areas of strength. Measuring a component's marginal contribution to an existing portfolio is better, but penalizes sets of strongly correlated algorithms, thereby obscuring situations in which it is essential to have at least one algorithm from such a set. This paper argues for analyzing component algorithm contributions via a measure drawn from coalitional game theory—the Shapley value—and yields insight into a research community's progress over time. We conclude with an application of the analysis we advocate to SAT competitions, yielding novel insights into the behaviour of algorithm portfolios, their components, and the state of SAT solving technology.

Introduction

Many important problems in AI are NP-complete but can still be solved very efficiently in practice by algorithms that exploit various kinds of instance structure. Different algorithms exploit such structure in subtly different ways, with the result that a given algorithm may dramatically outperform another on one problem instance, but the situation may be reversed on another. Algorithm portfolios (Huberman, Lukose, and Hogg 1997; Gomes and Selman 2001) are sets of algorithms that complement each other across an instance distribution. There has been much recent interest in algorithm portfolios, fueled by practical successes in SAT (Nudelman et al. 2003; Xu et al. 2008), CSP (O'Mahony et al. 2008), AI planning (Helmert, Röger, and Karpas 2011), Max-SAT (Malitsky, Mehta, and O'Sullivan 2013), QBF (Pulina and Tacchella 2009), ASP (Gebser et al. 2011), and many other problems (see, e.g., Kotthoff, 2014). Portfolios can run all algorithms in parallel, can use learned models to select an algorithm on a per-instance basis, or can employ variations of these ideas.

Regardless of how a portfolio is turned into an actual solver, it is often important to understand how much each component algorithm contributes to the success of the portfolio, both in order to winnow down the portfolio and to learn more about the algorithms themselves. This is also relevant in competitions that determine the current state of the art; the competition ranking and scores alone are often insufficient to adequately quantify the extent to which an algorithm contributes to the state of the art.

Perhaps the most natural way to assess the contribution of an algorithm is by simply assessing its standalone performance. More formally, let X denote a fixed set of instances of a given problem, $\mathcal{A} = \{i\}_{i=1}^{n}$ denote the set of available algorithms, and $A \subseteq \mathcal{A}$ denote an algorithm portfolio. Moreover, let $\mathsf{perf}(A)$ denote the performance achieved by leveraging the complementary strengths of the algorithms in A (e.g., by evaluating the performance on X of executing algorithm selection, or running all algorithms in parallel), and let $\mathsf{contr}(i, A)$ denote the contribution of i to A. Now if this contribution is evaluated solely based on the standalone performance of i, then we obtain the following measure of contribution:

$$\mathsf{contr}_{\mathrm{s}}(i, A) = \mathsf{perf}(\{i\}),$$

where s means *standalone*. While this measure is easy to compute, it fails to consider synergies in a portfolio and hence can give too much credit to strong but highly correlated algorithms. Furthermore, this measure fails to reward algorithms that perform poorly overall, but dramatically outperform other algorithms in A on some nontrivial subset of X (in a competition setting this means that the winner may perform very poorly on this subset). Motivated by these issues, Xu et al. (2012) argued that each algorithm i should be evaluated in terms of its *marginal contribution* to a given portfolio A. That is,

$$\mathsf{contr}_{\mathrm{m}}(i, A) = \mathsf{perf}(A) - \mathsf{perf}(A \setminus \{i\}),$$

where m stands for *marginal contribution*. The authors also included an application to SAT; Amadini, Gabbrielli, and Mauro (2013) conducted an analogous evaluation of CSP solvers. While the measure $\mathsf{contr}_{\mathrm{m}}$ addresses the problems discussed above, it raises some (arguably less severe) problems of its own—notably that it penalizes correlated component algorithms. In the most extreme case, a very important solver becomes completely unimportant if two identical versions are present in the portfolio, because neither makes a marginal contribution given the presence of the other.

In this work, we argue that the problem of assessing an algorithm's contribution benefits from being seen through the lens of coalitional (or "cooperative") game theory—a branch of microeconomic theory that studies the formation of coalitions and the ways in which they might divide rewards. One basis for such a division is *fairness*: we seek to reward players according to their contribution to the overall success of the coalition. The canonical measure of fairness in this sense is the *Shapley value* (Shapley 1953). Modelling portfolios of algorithms as coalitions—with players representing algorithms and the coalition's reward representing portfolio performance—we argue that the contribution of each algorithm should be measured according to the Shapley value. We apply the methodology to four SAT competitions (SAT Competitions Website 2014) and demonstrate that the Shapley value provides a more nuanced and useful assessment of component algorithm contributions. Notably, we identify cases in which an algorithm with top performance according to previous measures makes only small contributions to a portfolio, while the top contributor is not identified by other measures. This gives us a clearer picture of the state of the art in SAT solving—it is not just the competition winner or even the top ranked solvers that define it, but also solvers who on their own perform poorly and may not even have very large marginal contributions to the portfolio of all solvers.

The Shapley value can also be useful for assessing which algorithms should be included in a portfolio, but does not constitute an automatic technique for doing so. Instead, building an actual portfolio with a machine learning model for selecting the algorithm to run on a problem instance also needs to take the performance of the selection model into account. In particular, it may not be beneficial to include all algorithms that make a positive Shapley contribution as the selection model may not be able to leverage this. Conversely, algorithms that make no contribution at all may be neutral from the perspective of performance if the model is able to avoid them reliably. Overall, a portfolio containing algorithms selected according to Shapley values will be weakly larger than one containing algorithms selected based on their marginal contributions, giving it the potential to achieve better performance. However, investigating this issue is beyond the scope of this paper.

Assessing Algorithms with the Shapley Value

Coalitional game theory considers how coalitions form and in what ways it makes sense for them to assign credit for the value they create. Formally, a coalitional game is defined as a pair (N, v), where $N = \{1, \ldots, n\}$ is a set of n *players*, and $v : 2^N \to \mathbb{R}$ is a *characteristic function* that maps each subset (or *coalition*) of players $C \subseteq N$ to a real number $v(C)$. This number is called the *value* of coalition C, which typically represents the reward that can be attained by the members of C when they work together and coordinate their activities. The coalition of all the players, N, is called the *grand coalition*. We can represent our setting as a coalitional game by setting $N = A$, where each solver corresponds to a distinct player, and defining $v(C) = \mathsf{perf}(C)$.

The Shapley Value

Coalitional game theorists often assume that the grand coalition forms, and then ask how the value of this coalition should be divided amongst the players. There are many ways of answering this question; such answers are called *solution concepts*. One desirable criterion is *fairness*: assessing the extent to which each player contributed to the coalition's value. The solution concept that is canonically held to fairly divide a coalition's value is called the *Shapley value* (Shapley 1953). Indeed, out of all possible solution concepts, the Shapley value uniquely satisfies a set of desirable properties that characterize fair division of coalitional value, and thus it has been applied very widely (see, e.g., the book by Solan, Zamir, and Maschler, 2013).

We now describe the intuition behind the Shapley value. Arguably, if players joined the grand coalition one by one in a fixed order, a reasonable way to assess the contribution of each player would be simply to compute that player's marginal contribution to those who arrived before him. For instance, given three players, and given the joining order: $(2, 3, 1)$, the contribution of player 1 is $\mathsf{perf}(\{2, 3, 1\}) - \mathsf{perf}(\{2, 3\})$, the contribution of 3 is $\mathsf{perf}(\{2, 3\}) - \mathsf{perf}(\{2\})$, and of 2 is $\mathsf{perf}(\{2\}) - \mathsf{perf}(\emptyset)$. The Shapley value effectively generalizes this idea to situations in which there is no fixed sequential joining order, computing the *average* marginal contribution of each player over all possible joining orders. Formally, let Π^N denote the set of all permutations of N, each representing a different joining order. Moreover, given a permutation $\pi \in \Pi^N$, let C_i^π denote the coalition consisting of all predecessors of i in π. That is, $C_i^\pi = \{j \in N : \pi(j) < \pi(i)\}$, where $\pi(i)$ denotes the position of i in π. The *Shapley value of player* $i \in N$ is then defined as:

$$\phi_i = \frac{1}{n!} \sum_{\pi \in \Pi^N} \left(v(C_i^\pi \cup \{i\}) - v(C_i^\pi) \right). \tag{1}$$

Observe that $v(C_i^\pi \cup \{i\}) - v(C_i^\pi)$ is the marginal contribution of i to C_i^π.

One can use the notion of joining orders to contrast the standalone and marginal contribution metrics with the Shapley value. In particular, given a solver, i, the standalone metric assumes a joining order in which i comes first, whereas the marginal contribution metric assumes a joining order in which i comes last. As such, both metrics fail to capture all interactions between i and other solvers. The Shapley value, on the other hand, explicitly considers all possible joining orders. Indeed, Equation (1) can be rewritten as

$$\phi_i = \frac{1}{n} \sum_{c=0}^{n-1} \frac{1}{\binom{n-1}{c}} \sum_{\substack{C \subseteq N \setminus \{i\}: \\ |C| = c}} \left(v(C \cup \{i\}) - v(C) \right), \tag{2}$$

showing that the Shapley value is simply an average of average marginal contributions over the possible coalitions of each size. This means that it directly incorporates both the standalone and the marginal contribution metrics—these are the first and last terms in the sum, respectively.

Now, consider the following four properties, which seem desirable for a contribution measure, contr, to satisfy. It turns out that they *characterize* the Shapley value, in the sense that the Shapley value is the only measure that satisfies them all.

(1) **Efficiency**: $\sum_{i \in A} \mathsf{contr}(i, A) = v(A)$. Informally, the measure exactly splits the overall value achieved by the portfolio among the solvers.

(2) **Dummy player**: If $v(C) = v(C \cup \{i\})$ for all $C \subseteq A$, then $\mathsf{contr}(i, A) = 0$. Informally, if a solver does not impact the performance of any portfolio, $P(C) : C \subseteq A$, then it receives a score of zero.

(3) **Symmetry**: If two solvers i and j are *identical* in that they make the same marginal contributions to every portfolio $C \subseteq A \setminus \{i, j\}$ (i.e., $v(C \cup \{i\}) = v(C \cup \{j\})$) then they receive the same score (i.e., $\mathsf{contr}(i, A) = \mathsf{contr}(j, A)$).

(4) **Additivity**: Consider two arbitrary performance measures, perf_1 and perf_2. Define a new performance measure: $\mathsf{perf}_{1+2}(C) = \mathsf{perf}_1(C) + \mathsf{perf}_2(C)$ for every $C \subseteq A$. Define $\mathsf{contr}_{\mathsf{perf}}(i, C)$ as our assessment of algorithm i's contribution to portfolio C given the performance measure perf. Then, $\mathsf{contr}_{\mathsf{perf}_{1+2}}(i, C) = \mathsf{contr}_{\mathsf{perf}_1}(i, C) + \mathsf{contr}_{\mathsf{perf}_2}(i, C)$ for every algorithm i and every $C \subseteq A$. This, along with other properties, imply linearity: i.e., multiplying the performance measure by a constant—as could occur, e.g., because of running experiments on hardware of different speed—does not affect the ranking of solvers.[1]

SAT Competition Analysis

We apply the methodology outlined above to the SAT competition (SAT Competitions Website 2014) to assess the state of the art in SAT solving, and specifically, to fairly quantify the contributions of the participating solvers to the state of the art. We conducted experiments on solvers and problem instances from all SAT competitions for which the required performance data is publicly available—the 2014, 2013, 2011 and 2009 competitions. The competitions consist of three *tracks* of problem instances: *random*, *crafted* and *application*. We analyze the most recent competition—2014—first and in most detail, as it best reflects the current state of the art in SAT solving.

We assess each algorithm's contribution to the *virtual best solver (VBS)*, an oracle that chooses the best algorithm for every problem instance and also represents the wall-clock performance of an (overhead-free) parallel portfolio. The VBS-based portfolio solver serves as a theoretical upper bound on any portfolio solver's performance.

Performance measure

In the SAT competition, solvers are ranked by the number of instances they solve. Ties are broken to prefer algorithms with lower runtimes. We unify these criteria into a single scoring function as follows. Let $\mathsf{score}_x(i)$ denote the score of an algorithm $i \in A$ on an instance $x \in X$:

$$\mathsf{score}_x(i) = \begin{cases} 0 & \text{instance } x \text{ not solved by } i \\ 1 + \frac{c-t}{|X| \cdot c \cdot |A|+1} & \text{otherwise} \end{cases} \tag{3}$$

where c is the captime for running an instance and t the time required to solve it. We chose the denominator such that contributions to the score through runtime can never add up to more than 1, meaning that runtime differences can never be more important than solving an additional instance.

The score of a coalition $C \subseteq A$ is

$$\mathsf{score}_x(C) = \max_{i \in C} \mathsf{score}_x(i), \tag{4}$$

or the "virtual best score." The performance of C on a set of instances is

$$\mathsf{perf}_X(C) = \sum_{x \in X} \mathsf{score}_x(C). \tag{5}$$

We define the characteristic function v to be score and omit the subscript denoting the instance set as it is defined by the context.

Importantly, we show in Appendix 1 (available at http://tiny.cc/shapley-aaai16) that the above characteristic function can be represented compactly as a marginal contribution network and thus admits polynomial-time computation of the Shapley value (Chalkiadakis, Elkind, and Wooldridge 2011; Ieong and Shoham 2005).

Note that the Shapley value does not require the characteristic function to have any specific properties; it can be applied to any domain with any performance measure.

2014 Competition

The random track of the 2014 SAT competition consisted of 225 hard uniform k-SAT instances; the crafted track contained 300 instances manually designed to be challenging for typical SAT solvers; and the application track comprised 300 instances originating from applications of SAT to real world problems (e.g., software and hardware verification, cryptography, planning). For each track, we consider both satisfiable and unsatisfiable instances. Different specialized solvers participated in each track (10, 33 and 32 solvers for the random, crafted and application tracks, respectively).

Random Track

We first consider the random track of the 2014 SAT Competition and look in detail at differences between the standalone performance, marginal value and Shapley value measures. All of the solvers submitted to this track are stochastic local search solvers except for `SGSeq`, which combines the local search solver `SATtime` and the DPLL-based complete solver `Glucose`.

Figure 1 contrasts the three different measures of algorithm contribution to the VBS-based solvers. The three columns show the relative positions of each algorithm according to the standalone performance, Shapley value, and marginal value measures respectively, with lines connecting the three contribution measures calculated for each solver.

[1] See Macho-Stadler, Perez-Castrillo, and Wettstein (2007) for more details on the linearity of the Shapley value.

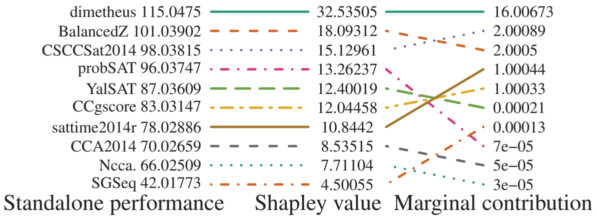

Figure 1: Standalone performance and contributions to the VBS portfolio for the random track 2014.

The results show that marginal contribution is very ineffective for distinguishing between algorithms; many solvers had very similar marginal contribution values. `dimetheus` beat all other algorithms under every contribution measure—in addition to very good standalone performance, it also complemented portfolios very well and was able to solve 16 more instances than all the other solvers combined. The set of instances it solved complements the sets of other solvers, and its Shapley value is almost twice as large as that of the second-ranked solver, while the difference in terms of standalone performance is much smaller.

In this case, the ranking under standalone performance is the same as the ranking under the Shapley value. However, the ranking under marginal contribution is significantly different. For example, `BalancedZ` loses out to `CSCCSat`, because it made smaller reductions in solution time on instances that were solved by other solvers, although it solved three more instances on its own. `sattime2014r` ranks fourth by marginal contribution, as it solved one instance that no other solver could tackle, while its standalone performance was poor, and it failed to complement the other solvers well. `SGSeq`, the only solver that employs non-local-search-based techniques, ranks last under standalone and Shapley value.

Crafted Track

Figure 2 shows the detailed Shapley value, standalone and marginal contribution results for the crafted track. In contrast to the random track, the rankings according to standalone performance and Shapley value are different. In particular, `glueSplit_clasp`, which has the best standalone performance, is ranked second after `SparrowToRiss` in terms of Shapley value. Interestingly, we can see from marginal contributions that `SparrowToRiss` did not solve any instances not solved by at least one other solver, whereas `glueSplit_clasp` solved two additional instances. The set of instances that `SparrowToRiss` solved is diverse and contains many instances that were solved by relatively few other solvers.

Again we see limitations of the marginal contribution measure: the values for many algorithms are exactly the same, with most solvers not making any marginal contribution at all, meaning that they do not solve any additional instances or solve any instances more quickly than the VBS portfolio consisting of all other solvers. In contrast, the Shapley value yields a more nuanced quantification of contributions, and some of solvers that make no marginal contribution make

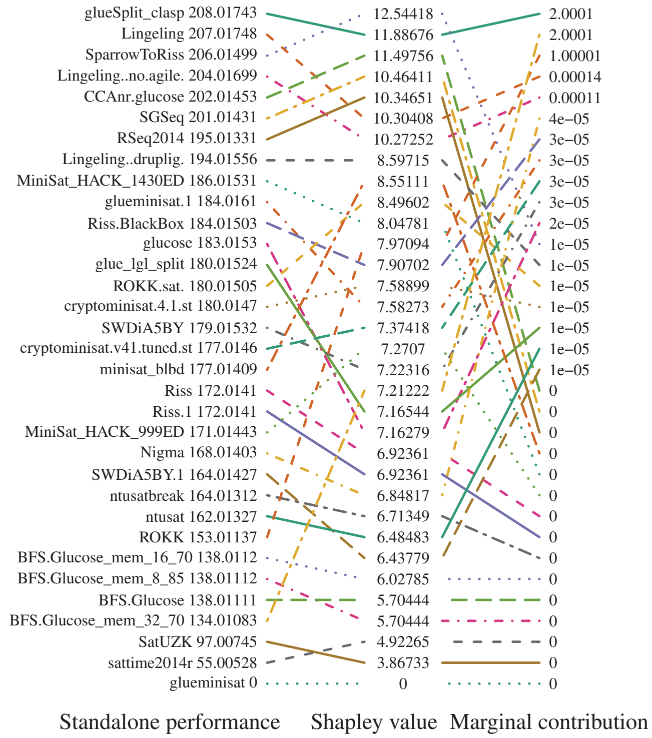

Figure 2: Standalone performance and contributions to the VBS portfolio for the crafted track 2014.

large Shapley value contributions.

The difference in Shapley value amongst the top-ranked solvers is much smaller than for the random track, indicating that the solvers show more similar performance, and the winner did not dominate as clearly as in the random track.

`Riss` and `Riss.1` exhibit identical standalone performance. Their marginal contributions are 0, indicating that they did not contribute anything to a portfolio containing all the other solvers. The Shapley value shows that they did contribute to smaller portfolios, and therefore suggests that at least one of them should be included in a portfolio.

Some solvers are ranked much lower in terms of Shapley value than in terms of standalone performance, notably `glucose` and `glue_lgl_split`. These solvers exhibited good performance in areas of the problem space where other solvers were good as well—they did not complement the state of the art well. Some other solvers, such as `ROKK` and `BFS.Glucose_mem_32_70`, have relatively high Shapley values but low standalone performance. These solvers complemented the state of the art more than they achieved on their own—combining them with other solvers in a portfolio produced something much more potent.

Many solvers that achieve high ranks in terms of Shapley value—e.g., `CCAnr.glucose`, `SGSeq` and `RSeq2014`—make no marginal contribution at all. This is clearly misleading; they have both very good standalone performance and complement many other portfolios well.

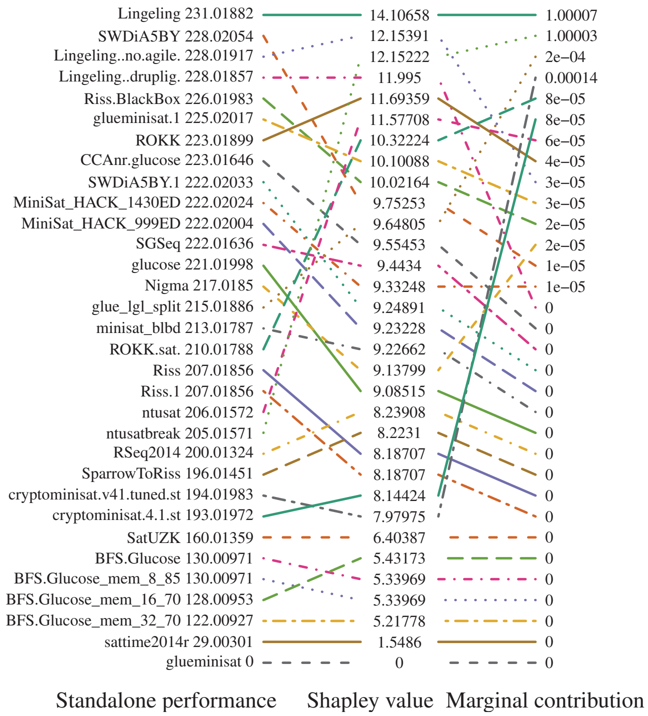

Figure 3: Standalone performance and contributions to the VBS portfolio for the application track 2014.

Application Track

Figure 3 shows results for the application track. Similar to the crafted track, the rankings under standalone performance and Shapley value are different, although the top solver, `Lingeling`, ranks highest under all three measures. The performance values indicate that it did not dominate the other solvers to the same extent as `dimetheus` did in the random track. Again, we see no marginal contribution from a large number of solvers, and almost no marginal contribution for a large fraction of the remainder.

The second and third-ranked solvers in terms of Shapley value, `Lingeling-no-agile` and `ntusatbreak`, have almost the same Shapley value. This is somewhat surprising, as one of the solvers solves 10% more instances. The reason for this lies in the fact that `ntusatbreak`, although weaker in terms of standalone performance, contributed somewhat more across the gamut of portfolios including other solvers from this track.

`SWDiA5BY`, on the other hand, shows very good standalone performance, but only mediocre Shapley value contribution. This indicates that the types of instances where this solver shone were already adequately covered by other solvers, and that it did not contribute much on instances on which other solvers did not perform well. `cryptominisat.v41.tuned.st` and `cryptominisat.4.1.st` are ranked towards the bottom in terms of standalone performance and Shapley value, but at the top in terms of marginal contribution. This indicates that

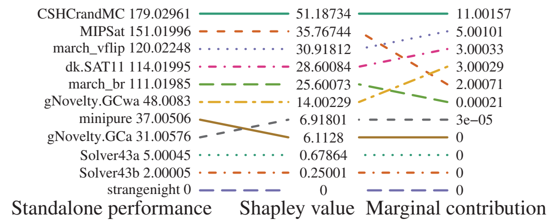

Figure 4: Standalone performance and contributions to the VBS portfolio for the random track 2013.

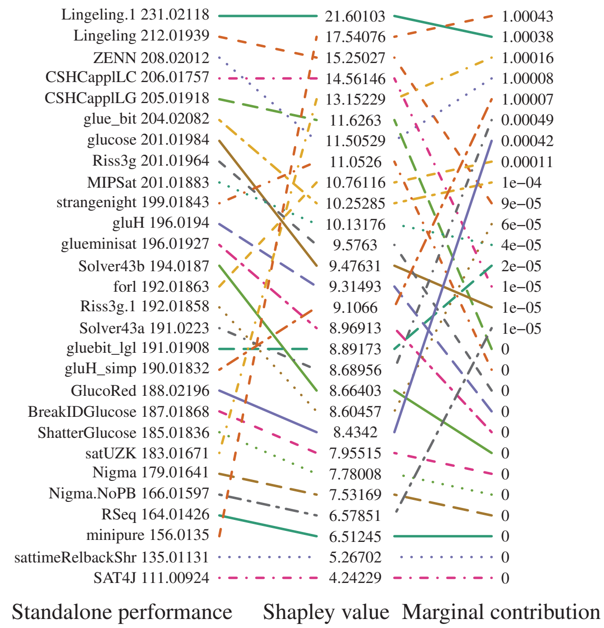

Figure 5: Standalone performance and contributions to the VBS portfolio for the application track 2013.

these solvers achieved a small performance improvement on a very small set of instances, but did not contribute meaningfully otherwise.

2009–2013 Competitions

We now turn to an analysis of performance data from the 2013, 2011 and 2009 Competitions. Our findings were similar to those for the 2014 data—in many cases, we saw significant rank changes under the Shapley value ranking as compared to standalone performance and marginal contribution. We do not show results for all years and all tracks due to lack of space, but report some of the most interesting results.

Portfolio solvers were not allowed to enter the most recent SAT competition, but did participate in earlier competitions. We observe that when they were submitted (`SATzilla`, `ppfolio`, `CSHC`, `Satisfiability Solver Selector`), they did not consistently the highest Shapley values. This may be surprising, considering the success of portfolio solvers in SAT competitions. However, there is a simple explanation:

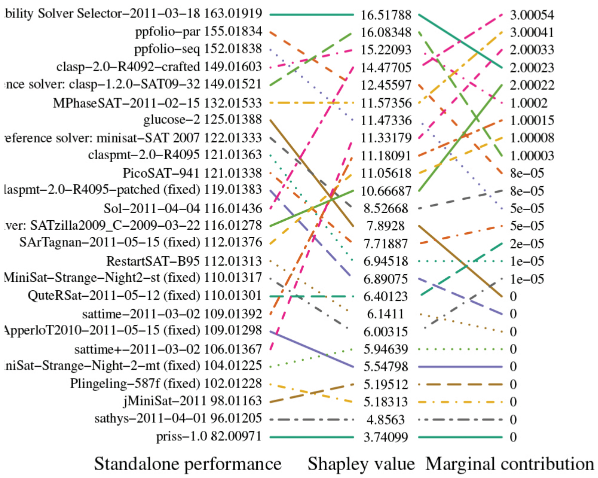

Figure 6: Standalone performance and contributions to the VBS portfolio for the random track 2011.

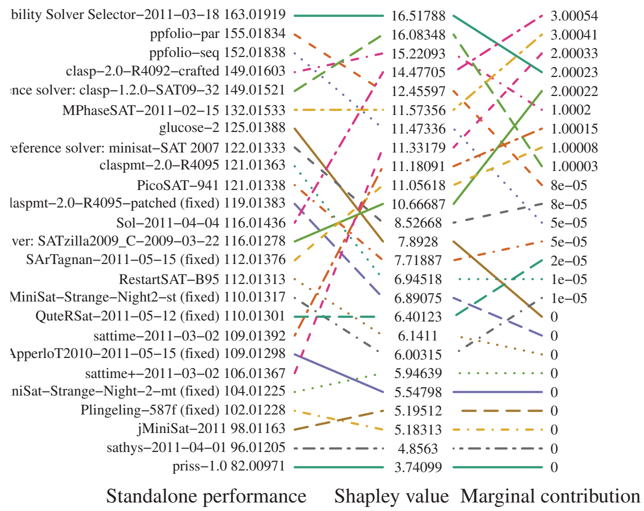

Figure 7: Standalone performance and contributions to the VBS portfolio for the crafted track 2011.

this occurred when very high-performing new algorithms were newly introduced and hence not included in portfolio submissions, which are only able to draw on previously published solvers. For example, the winner of the 2011 random track (Figure 6), `borg-sat-11.03.02-seq-random`, did not participate in 2009 and could therefore not be included in the portfolio solver `Satisfiability Solver Selector`, which placed second. `glucose-2` beat the portfolio solvers in the 2011 application track (Figure 8), as only its predecessor, `glucose-1` participated in 2009.

The `minipure` solver in the 2013 application track (Figure 5) achieved the most notable difference between standalone and Shapley value rankings: on its own, it ranks third last; in terms of Shapley value, it ranks second! We can conclude that it complemented portfolios of other solvers extremely well, even though on its own, it was only able to solve a small number of instances.

Multi-Year Analysis

Considering multiple years of SAT competitions allows us to analyse how the state of the art has evolved. We illustrate the insights that can be gained from considering multiple years by a few examples and defer a more in-depth analysis to an extended version of this paper due to limited space.

The 2009 and 2011 competitions include the top-ranked solvers from the respective previous competitions. In the 2011 crafted track (Figure 7), for example, the highest-ranked solver from 2009, `clasp-1.2.0`, achieves only fifth place in terms of standalone performance, and its rank in terms of marginal contribution is even lower. However, the Shapley value contribution puts it in second place, indicating that it still contributes substantially to the state of the art.

Conclusions

We have introduced a measure for the contribution of a solver to the performance of a portfolio-based approach, such as an algorithm selector or a parallel portfolio. Our measure is based on a foundational concept from coalitional game theory, the Shapley value. We have shown how the Shapley value addresses weaknesses of two established measures, standalone and marginal contribution, and permits a more nuanced analysis that produces interesting additional insights. Although we have illustrated its application in the context of SAT solving (arguably the area in which portfolio-based techniques have had the largest impact), we expect it to be equally useful for the analysis of component contributions to portfolio-based solvers for other problems. Because it does a better job of surfacing solvers that introduce novel, complementary strategies, we argue that the Shapley value should be used as a scoring function for SAT competitions and other competitive solver evaluations.

Acknowledgments

Kevin Leyton-Brown, Alexandre Fréchette and Lars Kotthoff were supported by an NSERC E.W.R. Steacie Fellowship; in addition, all of these, along with Holger Hoos, were supported under the NSERC Discovery Grant Program and Compute Canada's Research Allocation Competition. Part of the work was done while Lars Kotthoff was at Insight Centre for Data Analytics, Cork, Ireland. Tomasz Michalak was supported by the European Research Council under Advanced Grant 291528 ("RACE").

References

Amadini, R.; Gabbrielli, M.; and Mauro, J. 2013. An empirical evaluation of portfolios approaches for solving CSPs. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 7874 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 316–324.

Chalkiadakis, G.; Elkind, E.; and Wooldridge, M. 2011. *Computational Aspects of Cooperative Game Theory*. Synthesis

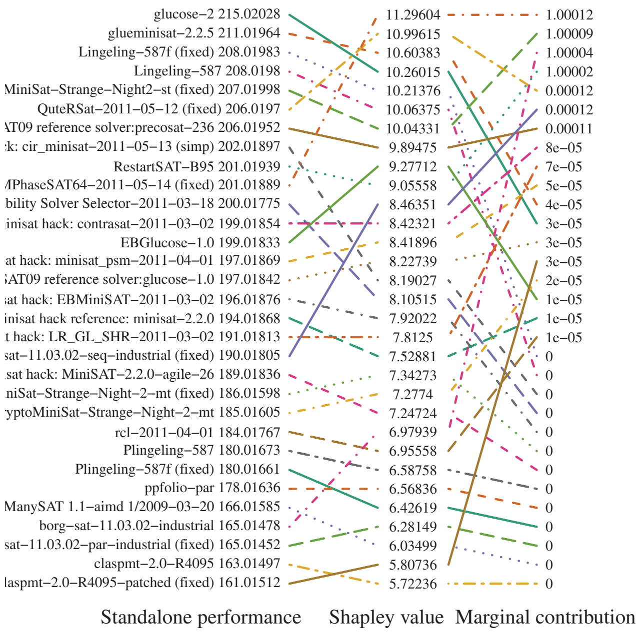

Figure 8: Standalone performance and contributions to the VBS portfolio for the application track 2011.

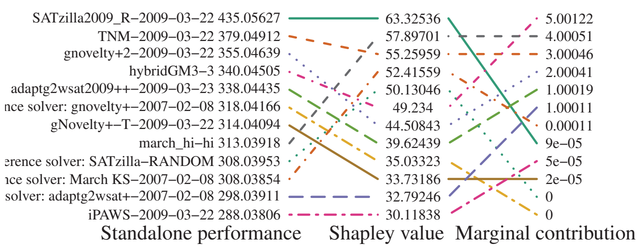

Figure 9: Standalone performance and contributions to the VBS portfolio for the random track 2009.

Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Gebser, M.; Kaminski, R.; Kaufmann, B.; Schaub, T.; Schneider, M. T.; and Ziller, S. 2011. A portfolio solver for answer set programming: preliminary report. In *11th International Conference on Logic Programming and Nonmonotonic Reasoning*, 352–357. Springer.

Gomes, C. P., and Selman, B. 2001. Algorithm portfolios. *Artificial Intelligence* 126(12):43–62.

Helmert, M.; Röger, G.; and Karpas, E. 2011. Fast downward stone soup: A baseline for building planner portfolios. In *ICAPS-2011 Workshop on Planning and Learning (PAL)*, 28–35.

Huberman, B. A.; Lukose, R. M.; and Hogg, T. 1997. An economics approach to hard computational problems. *Science* 275:51–54.

Ieong, S., and Shoham, Y. 2005. Marginal contribution nets: a

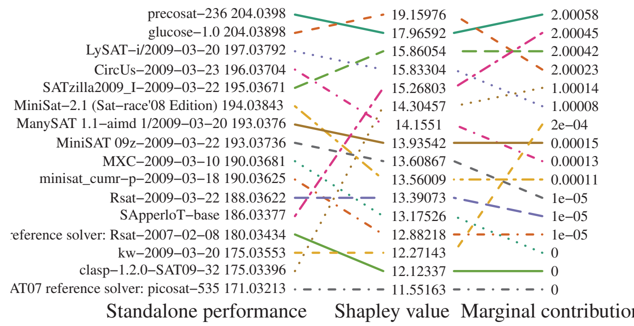

Figure 10: Standalone performance and contributions to the VBS portfolio for the application track 2009.

compact representation scheme for coalitional games. In *EC '05: Proceedings of the Sixth ACM Conference on Electronic Commerce*, 193–202.

Kotthoff, L. 2014. Algorithm selection for combinatorial search problems: A survey. *AI Magazine* 35(3):48–60.

Macho-Stadler, I.; Perez-Castrillo, D.; and Wettstein, D. 2007. Sharing the surplus: An extension of the Shapley value for environments with externalities. *Journal of Economic Theory* 135(1):339–356.

Malitsky, Y.; Mehta, D.; and O'Sullivan, B. 2013. Evolving instance specific algorithm configuration. In *The Sixth Annual Symposium on Combinatorial Search*.

Nudelman, E.; Leyton-Brown, K.; Andrew, G.; Gomes, C.; McFadden, J.; Selman, B.; and Shoham, Y. 2003. Satzilla 0.9. Solver description, International SAT Competition.

O'Mahony, E.; Hebrard, E.; Holland, A.; Nugent, C.; and O'Sullivan, B. 2008. Using case-based reasoning in an algorithm portfolio for constraint solving. In *Proceedings of the 19th Irish Conference on Artificial Intelligence and Cognitive Science*.

Pulina, L., and Tacchella, A. 2009. A self-adaptive multi-engine solver for quantified boolean formulas. *Constraints* 14(1):80–116.

2014. SAT Competitions Website. http://www.satcompetition.org.

Shapley, L. S. 1953. A value for n-person games. In *Contributions to the Theory of Games, volume II*. Princeton University Press. 307–317.

Solan, E.; Zamir, S.; and Maschler, M. 2013. *Game Theory*. Cambridge University Press.

Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2008. SATzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research* 32:565–606.

Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2012. Evaluating component solver contributions to portfolio-based algorithm selectors. In *Proceedings of the 15th international conference on Theory and Applications of Satisfiability Testing*, SAT'12, 228–241. Springer-Verlag.