Department of Computer Science

A Game Theoretic Approach to Measure Contributions in Algorithm Portfolios

Talal Rahwan, **University of Southampton, UK**
Tomasz P. Michalak **University of Oxford, UK**

CS-RR-13-11

Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford, OX1 3QD

A Game Theoretic Approach to Measure Contributions in Algorithm Portfolios

Talal Rahwan[1] and Tomasz Michalak[2]

[1]School of Electronics and Computer Science, University of Southampton, UK
[2]Department of Computer Science, University of Oxford, UK

(28 August, 2013)

Abstract

Algorithm Portfolios [6, 5] have attracted significant attention in Artificial Intelligence research, due to their ability to exploit the complementary strengths that often exist among different algorithms. In this article, we address the following natural question: *How do we measure the contribution that an algorithm makes to a portfolio of algorithms?* We show that the solution proposed in the literature to answer this question is inadequate. We then propose the use of the *Shapley value*—a well-known concept in cooperative game theory—and show that it provides the correct answer. We discuss the potential impact and insights that the use of this measure could bring to the community.

1 Introduction

One of the main questions that are addressed in Artificial Intelligence research is how to develop new algorithms to solve various problems. When dealing with any such problem, a general trend in the literature is to focus on certain characteristics that exist in some instances of that problem. In so doing, it is usually possible to develop a new algorithm aimed at exploiting those characteristics, thus outperforming other algorithms whenever the problem instance happens to exhibit those characteristics. By repeating this algorithm-development process over several years, we end up with a set of algorithms that have (at least sometimes) complementary strengths. As a result, a new trend has emerged in recent years; instead of developing new algorithms, the aim is to develop what is known as an *Algorithm Portfolio* [6, 5], which is basically a collection of existing, complementary algorithms (often referred to as *components*). Then, given a problem instance, a portfolio either selects a single algorithm to solve it, or runs multiple algorithms (either in parallel on multiple processors, or interleaved on a single processor according to some schedule). An algorithm portfolio is particularly useful when solving a set of diverse problem instances, where no single algorithm dominates the others for all instances. One success story from this line of research is the development of `SATzilla` [12]—the Portfolio-based solver for the *Boolean satisfiability problem (SAT)*, which has earned its developers several gold medals in the SAT competitions. Other successful portfolios for SAT include `3S` [7] and `ppfolio` [10].

For problems (such as SAT) were multiple algorithms, and multiple algorithm portfolios, are available, the following natural question arises: *what is the current state of the art (SOTA) for solving that problem?* One way to answer this question is to compare individual algorithms, and then select the best one based on some criteria (e.g., run time, or number of instances solved). Another comparison that one can make is between different algorithm portfolios, to find out the best available portfolio. The very reason for holding the SAT competition, for example, is to carry out the two aforementioned comparisons in a standardized fashion.

Xu et al. [13] highlighted the need for a third comparison in order to provide a complete answer to the aforementioned question. This comparison is *between individual algorithms based on their contributions towards successful portfolios.* For instance, an algorithm that focuses on hard but rare instances might not stand out when compared against other algorithms given a wide variety of instances. When included into a portfolio, such an algorithm might contribute significantly towards the portfolio's success, but might not receive the recognition it deserves, unless it is compared against other components in the portfolio, based on the contribution that each has made.

To carry out the aforementioned (third) comparison, one needs to first be able to answer the following, seemingly-easy question: *what is the contribution that a particular algorithm has made to the outcome of an algorithm portfolio?* An accurate answer would clearly provide valuable information that can be beneficial in various ways. For instance, it can provide a better understanding of the synergistic effect that results from combining different algorithms. It can also provide a basis based on which the algorithms in the portfolio can be modified, e.g., by removing the one(s) that contributed the least, or keeping the one(s) that contributed the most.

In this article, we show that the only attempt to measure the contribution of an algorithm to a portfolio, due to Xu et al. [13], is inadequate (see Section 3 for more details). Motivated by this observation, we propose an adequate measure by borrowing concepts from cooperative game theory—a branch of microeconomics that studies the interactions among different *"players"* in scenarios where cooperation is possible through binding agreements. One of the main questions addressed in this area of research is how to divide the reward from cooperation among players so as to meet certain desirable criteria. One such criterion is *fairness*: how well does each player's reward reflect its contribution? To date, the best known answer to this question is the *Shapely value* [11]. This solution concept, now one of the fundamental results in cooperative game theory, specifies how the contribution that each player has made should be measured when those players cooperate.[1] In subsequent sections, we will present a formal definition of the Shapley value, and explain of the intuition behind it.

We propose to model "portfolios" of algorithms as "coalitions" in a cooperative game, where each "player" represents an algorithm, and the improvement in performance is represented as the "reward" attained when those players "cooperate". By drawing parallels between algorithm portfolios and cooperative game

[1]This solution concept is named after Lloyd Shapley—the 2012 Nobel Prize winner who proposed this concept in 1953.

theory, we obtain an accurate measure of the contribution that each algorithm has made to improvement in performance; this measure is the Shapley value.

The remainder of this article is structured as follows. Section 2 is intended to make the reader familiar with some necessary concepts from cooperative game theory. Section 3 highlights the shortcomings of the existing approach for measuring the contribution of an algorithm to a portfolio. Section 4 presents the Shapley value-based approach in detail, and outlines its advantages. Section 5 discusses the potential impact that this approach could have on the community. Finally, Section 6 outlines some potential future directions.

2 Preliminaries

In this section, we briefly introduce some necessary definitions and concepts from cooperative game theory.

A *characteristic function game*, G, is given by a pair (P, v), where $P = \{p_1, \dots, p_n\}$ is a finite set consisting of n *players*, and $v : 2^P \to \mathbb{R}$ is a *characteristic function* that maps each subset (or *coalition*) of players, $C \subseteq P$, to a real number, $v(C)$. This number is referred to as 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. In this context, the terms *"payoff"* or *"utility"* are often used in the literature instead of the term *"reward"*.

A *coalition structure*, $CS \subseteq 2^P$, is a partition of the set of players, i.e., it is a set of disjoint and exhaustive coalitions. We will denote the set of all possible coalition structures as $\mathcal{CS}^P$.

An *outcome* of a game is a pair, $(CS, \mathbf{x})$, where $CS \in \mathcal{CS}^P$ is a coalition structure, and $\mathbf{x} = (x_1, \dots, x_n)$ is a *payoff vector*, which specifies how the value of each coalition $C \in CS$ is distributed among its members. More specifically, x_i is the *payoff* of player p_i in CS, such that $x_i \geq 0$ for all $i = 1, \dots, n$, and $\sum_{p_i \in C} x_i = v(C)$ for all $C \in CS$. In other words, an outcome specifies, for every player, the coalition to which it belongs (i.e., the players with which it cooperates and coordinates its activities), and the reward that this player receives from such cooperation.

A *solution concept* specifies the set of outcomes that meet certain criteria. One desirable criterion that is often sought after is *fairness*—how well does each player's payoff reflect the contribution that this player has made to its coalition. Here, the main question is *how to measure this contribution*, i.e., how to measure the extra utility that the player's membership brings to its coalition. In this context, the *Shapley value* [11] is the best-known solution concept (to date) that aims at capturing this notion of fairness in cooperative game theory. We postpone the description of how the Shapley Value is computed to Section 4.

3 Limitations of the Existing Measure

Xu et al. [13] were the first to highlight the importance of measuring the contributions that individual algorithms make towards the outcome of an algorithm portfolio. They argue that the algorithm's contribution should be measured as the algorithm's *marginal contribution* to the portfolio—the difference between the portfolio's performance including the algorithm and the portfolio's performance excluding it. This measure was later on used by Amadini, Gabbrielli, and Mauro [1] to evaluate the contributions of algorithms to portfolios that solve constraint satisfaction problems (CSPs).

Next, we will show that the above method is not an adequate measure of contributions. We will show this by analysing a sample cooperative game, $G = (P, v)$, where the set of players is $P = \{p_1, \dots, p_4\}$, and the characteristic function—the function that associates a value to every possible coalition—is v, defined as follows:

$$\begin{array}{llll}
v(\{p_1\}) = 80 & v(\{p_1,p_2\}) = 110 & v(\{p_1,p_2,p_3\}) = 180 & v(P) = 200 \\
v(\{p_2\}) = 80 & v(\{p_1,p_3\}) = 130 & v(\{p_1,p_2,p_4\}) = 180 & \\
v(\{p_3\}) = 90 & v(\{p_1,p_4\}) = 120 & v(\{p_1,p_3,p_4\}) = 170 & \\
v(\{p_4\}) = 90 & v(\{p_2,p_3\}) = 130 & v(\{p_2,p_3,p_4\}) = 170 & \\
 & v(\{p_2,p_4\}) = 120 & & \\
 & v(\{p_3,p_4\}) = 110 & &
\end{array}$$

The interpretation of this game from our algorithm portfolio perspective is as follows. Each "player" in the game represents an algorithm, and the set of players, P, represents the algorithm portfolio. The "value" of a coalition (i.e., a subset of algorithms), $C \subseteq P$, represents some performance measure of the portfolio that consists of the members of C. For instance, $v(\{p_2, p_3\})$ could represent the number of instances that can be solved by a portfolio consisting of algorithms p_2 and p_3.

As can be seen, if the *grand coalition* (i.e., the biggest possible coalition, P) forms, then the payoff would be 200 units of utility. Let us determine how many of these 200 units resulted from the membership of, say, p_1 in P. One way to do this is by using the method proposed by Xu et al. [13], which is to compare the payoff of P (which is 200) with the payoff of $P \setminus \{p_1\}$ (which is 170). The conclusion drawn from this conclusion, according to Xu et al. , is that the membership of p_1 in P brings an extra 30 units of utility. Following a similar approach, we conclude that players p_2, p_3, p_4 bring an extra $30, 20, 20$ units of utility by being members of P, respectively. This implies that the total utility of P is the sum of utilities brought by each member, i.e., $30 + 30 + 20 + 20 = 100$. However, this contradicts what we know about $v(P)$ being equal to 200. The reason behind this contradiction is that, when computing the contribution of p_i as per the aforementioned approach, it was implicitly *assumed that p_i was the last player to join P.* More specifically, when computing how much p_1 has contributed towards the 200 units attained by P, we compared it against the utility attained by $P \setminus \{p_1\}$, and concluded that p_1's contribution is: $v(\{p_1, p_2, p_3, p_4\}) - v(\{p_2, p_3, p_4\}) = 200 - 170 = 30$. This implies that, before p_1 has joined P, the players p_2, p_3, p_4 were already in P, and so P was already capable of attaining 170 units. In this case, indeed having p_1 join P would only bring an extra 30 units of utility. By following the same approach for every player, we are basically assuming that every one of the four players has joined P last, which is simply an impossible joining order (this is because, by definition, only one player can join last). Next, we show how this problem can be tackled, by considering only joining orders that are possible.

4 Shapley Value-based Approach

To better quantify the extra utility that the membership of p_i brings to $v(P)$, we use the *Shapley value* [11]. According to this solution concept, the contribution of a player to $v(P)$ is measured *while taking into consideration all possible joining orders.* To present the formal definition of the Shapley value, we need some additional notation. Let Π^P denote the set of all permutations of P, i.e., one-to-one mappings from P to itself. Basically, every permutation, $\pi \in \Pi^P$, represents a possible joining order, where the first player in π is the first to join P, the second in π is the second to join P, and so on. For instance, $\pi = \langle p_3, p_4, p_1, p_2 \rangle$ represents the joining order where p_3 joins first, followed by p_4, then p_1 then p_2. Now, for any arbitrary permutation, $\pi \in \Pi^P$, let C_i^π denote the coalition consisting of all predecessors of p_i in π. More formally, if we denote by $\pi(p_i)$ the location of p_i in π, then $C_i^\pi = \{p_j \in P : \pi(p_j) < \pi(p_i)\}$. Now, we are ready to introduce the notion of *marginal contribution* from a game theoretic perspective. Basically, the marginal contribution of a player p_i is defined with respect to a permutation π in a game $G = (P, v)$ as follows:

$$\Delta_\pi^G(p_i) = v(C_i^\pi \cup \{p_i\}) - v(C_i^\pi)$$

In other words, according to a given joining order, π, the marginal contribution of p_i is the extra utility that p_i brings when it joins the players that precede it in π. Clearly, the marginal contribution of p_i is influenced by the joining order. For instance, given $\pi = \langle p_3, p_4, p_1, p_2 \rangle$, we have $\Delta_\pi^G(p_1) = v(\{p_1, p_3, p_4\}) - v(\{p_3, p_4\}) = 60$. On the other hand, given $\pi = \langle p_3, p_1, p_2, p_4 \rangle$, we have $\Delta_\pi^G(p_1) = v(\{p_1, p_3\}) - v(\{p_3\}) = 40$.

Having introduced the notion of marginal contribution, we can now introduce the Shapley value. Basically, it is the solution concept based on which the payoff of a player p_i is the *average marginal contribution* over all possible joining orders. Formally:

Definition 1 *Given a characteristic function game $G = (P, v)$, the* Shapley value of a player[2] *$p_i \in P$ is denoted by $\phi_i(G)$ and is given by*

$$\phi_i(G) = \frac{1}{n!} \sum_{\pi \in \Pi^P} \Delta_\pi^G(p_i). \tag{1}$$

Computing the Shapley value of a given player is hard. This is because of the need to consider all possible joining orders, which are $n!$ in total (given n

[2]Observe that if we compute the payoff vector, $\mathbf{x}$, according to the Shapley value, then x_i (i.e., the payoff of player p_i in $\mathbf{x}$) is referred to as the "Shapley value of p_i".

players). This computation can be made a little easier based on the following observation: For any permutation, π, and any player, p_i, the marginal contribution of p_i depends solely on the identities (and not the order) of the players that appear before p_i, and those that appear after it, in π. For instance, the marginal contribution of p_1 is the same in: $\langle p_3, p_4, p_1, p_2 \rangle$ and $\langle p_4, p_3, p_1, p_2 \rangle$. For all such similar permutations, it suffices to compute the marginal contribution once, and multiply it by the number of those permutations. Formally, the following alternative formula can be used instead of (1) to compute the Shapley value of p_i:

$$\phi_i(G) = \sum_{C \subseteq P \setminus \{p_i\}} \frac{|C|!\ (|P| - |C| - 1)!}{|P|!} \big(v(C \cup \{p_i\}) - v(C)\big) \qquad (2)$$

This formula requires considering all subsets of $P \setminus \{p_i\}$, which are 2^{n-1} in total. While this number is significantly smaller than $n!$—the number of permutations to be considered when using (1)—this number is still exponential in n, i.e., the Shapley value is still hard to compute given a large number of players. Fortunately, it is possible to approximate the Shapley value by sampling from those 2^{n-1} subsets, instead of considering them all. Such an approximation often yields values that are sufficiently close to the actual Shapley value (see, e.g., [2]). It is also possible to bound the estimation error when using sampling-based methods [8]. One can also speed up those methods by parallelizing. Specifically, when estimating the Shapley value of a player $p_i \in P$, multiple processors can each simultaneously sample from the space of possible subsets of $P \setminus \{p_i\}$, and can compute the marginal contribution of p_i to each sample. Then, each processor returns the average of all the marginal contributions that it has computed. Finally, the averages returned by different processors can all be aggregated in a straightforward manner to obtain the estimated Shapley value.

The Shapley value has many attractive properties. In what follows, we list four of them.

(1) **Efficiency**: all the profit earned by the players in P is distributed among them. Formally, given a characteristic function game $G = (P, v)$, we have: $\sum_{p_i \in P} \phi_i(G) = v(P)$;

(2) **Dummy player**: The Shapley value does not allocate any payoffs to *dummy* players, i.e., players who do not contribute to any coalition. Formally, given a characteristic function game $G = (P, v)$, if $v(C) = v(C \cup \{p_i\})$ for every $C \subseteq A$, then $\phi_i(G) = 0$;

(3) **Symmetry**: Given a characteristic function game $G = (P, v)$, we say that players p_i and p_j are *symmetric* in G if $v(C \cup \{p_i\}) = v(C \cup \{p_j\})$ for every $C \subseteq A \setminus \{p_i, p_j\}$. The Shapley value of symmetric players is equal;

(4) **Additivity**: Finally, consider a group of players P that is involved in two coalitional games G' and G'', i.e., $G' = (P, v')$, $G'' = (P, v'')$. The *sum* of G' and G'' is a coalitional game $G^+ = G' + G''$ given by $G^+ = (P, v^+)$, where for every coalition $C \subseteq A$ we have $v^+(C) = v'(C) + v''(C)$. The Shapley

value of a player p_i in G^+ is the sum of its Shapley values in G' and G''. This property implies the linearity of the Shapley value; if the characteristic function is multiplied by a constant, the Shapley values will simply be scaled by that constant. In other words, multiplying the performance measure by a constant does not affect the ranking of algorithms.

Interestingly, the Shapley value is the only payoff division scheme that has all of the above four properties [11]. In other words, if we view properties (1)–(4) as axioms, then these axioms characterize the Shapley value. This means that, unless the Shapley value is used, one has to sacrifice at least one of these important properties.

5 Discussion

Commenting on scoring schemes in SAT competitions, Gelder et al. [4] say:

"the major impact of being ranked among the best solvers is beneficial both for academic and industrial competitors. As a consequence, the scoring scheme of the competition needed some more formal basis."

This statement emphasizes the importance of using the Shapley value, as it specifies a principled way for measuring the contribution that an algorithm has made to a portfolio of algorithms.

One of the advantages of using the Shapley value is that it can be used with any performance measure. For example, the performance of a portfolio can be measured by the number of benchmarks it solves, or the CPU time required to solve a given benchmark, or some combination of the two. Here, one can use actual values, or predicted ones.

By obtaining a ranking of different algorithms in a portfolio (based on their contributions), it may be possible to iteratively modify the composition of the portfolio in a principled manner, e.g., by removing the k least contributing algorithms and/or keeping the k' most contributing ones (a similar algorithm was developed for *feature selection*, see [3]). Another possibility for using the Shapley-based ranking is to adjust any portfolio in which algorithms are interleaved on a single processor. In particular, it is possible to adjust the time slices in the portfolio schedule such that algorithms with greater contributions are allocated greater time.

Incorporating a Shapley value-based scoring scheme in competitions (e.g., the SAT competition) may encourage researchers to develop algorithms that are not necessarily the best on their own, but are capable of complementing other existing algorithms. This could be done, for instance, by focusing on rare but hard problem instances, without having to worry about whether the algorithm would receive the recognition it deserves from the community. One way to implement this scoring scheme is as follows: Once the winning portfolios are determined in the competition, it is possible to compute the Shapley value for each algorithm, and award the one that contributed the most. Another way to implement this scheme is to propose a new category in the SAT competition, where all participating algorithms are combined into one big portfolio. The organizers could agree in advance on how this portfolio manages its components, and may provide a detailed description online to be available for participants in advance. The results could even encourage the design of new benchmarks to be used in the competition.

6 Future Work

For future work, we would like to run experiments on various instances of the SAT problem, where the Shapley value is used to measure the contributions of algorithms to existing portfolios. We will analyse the resulting rankings, in the hope to obtain a better understanding of the complementarity between different algorithms. We will also analyse the corresponding cooperative game, e.g., to determine whether it is *monotonic*, *submodular*, *subadditive*, etc. This could help identify the most efficient way for approximating the Shapley value. It is also interesting to evaluate how the approximated contributions converge to the actual ones over the runtime of a sampling-based method, e.g., to quantify the number of samples needed before obtaining satisfactory error bounds.

References

[1] Amadini, R.; Gabbrielli, M.; and Mauro, J. 2013. An empirical evaluation of portfolios approaches for solving csps. In Gomes, C., and Sellmann, M., eds., *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.

[2] Castro, J.; Gómez, D.; and Tejada, J. 2009. Polynomial calculation of the shapley value based on sampling. *Computers* & *OR* 36(5):1726–1730.

[3] Cohen, S.; Ruppin, E.; and Dror, G. 2005. Feature selection based on the shapley value. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, 665–670. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[4] Gelder, A. V.; Berre, D. L.; Biere, A.; Kullmann, O.; and Simon, L. 2005. Purse-based scoring for comparison of exponential-time programs. In *Proc. of SAT-05*.

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

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

[7] Kadioglu, S.; Malitsky, Y.; Sabharwal, A.; Samulowitz, H.; and Sellmann, M. 2011. Algorithm selection and scheduling. In Lee, J., ed., *Principles and*

Practice of Constraint Programming CP 2011, volume 6876 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 454–469.

[8] Maleki, S.; Tran-Thanh, L.; Hines, G.; Rahwan, T.; and Rogers, A. 2013. Bounding the estimation error of sampling-based shapley value approximation with/without stratifying. Technical report.

[9] Nikolić, M.; Marić, F.; and Janičić, P. 2009. Instance-based selection of policies for sat solvers. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, SAT '09, 326–340. Berlin, Heidelberg: Springer-Verlag.

[10] Roussel, O. 2011. Description of ppfolio. `www.cril.univ-artois.fr/~roussel/ppfolio/solver1.pdf`. Solver description, last visited on May, 2012.

[11] Shapley, L. S. 1953. A value for n-person games. In Kuhn, H. W., and Tucker, A. W., eds., *Contributions to the Theory of Games, volume II*. Princeton University Press. 307–317.

[12] Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2008. Satzilla: Portfolio-based algorithm selection for sat. *J. Artif. Intell. Res. (JAIR)* 32:565–606.

[13] Xu, L.; Hutter, F.; Hoos, 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. Berlin, Heidelberg: Springer-Verlag.