Quantifying Algorithmic Improvements over Time

Lars Kotthoff[1], **Alexandre Fréchette**[2], **Tomasz Michalak**[3],
Talal Rahwan[4], **Holger H. Hoos**[5,2], **Kevin Leyton-Brown**[2]
[1]University of Wyoming,
[2]University of British Columbia,
[3]University of Warsaw,
[4]Masdar Institute of Science and Technology,
[5]Universiteit Leiden

Abstract

Assessing the progress made in AI and contributions to the state of the art is of major concern to the community. Recently, Fréchette *et al.* [2016] advocated performing such analysis via the Shapley value, a concept from coalitional game theory. In this paper, we argue that while this general idea is sound, it unfairly penalizes older algorithms that advanced the state of the art when introduced, but were then outperformed by modern counterparts. Driven by this observation, we introduce the *temporal Shapley value*, a measure that addresses this problem while maintaining the desirable properties of the (classical) Shapley value. We use the temporal Shapley value to analyze the progress made in (i) the different versions of the Quicksort algorithm; (ii) the annual SAT competitions 2007–2014; (iii) an annual competition of Constraint Programming, namely the MiniZinc challenge 2014–2016. Our analysis reveals novel insights into the development made in these important areas of research over time.

1 Introduction

Heuristic algorithms have played a key role in advancing many areas of AI by providing a practical way to identify reasonable (albeit not necessarily optimal) solutions to NP-hard problems. Such algorithms typically exploit the structure of a particular problem instance in order to cut down large areas of the search space and quickly narrow down the set of possible solutions. Such algorithms often exhibit significant performance variation across problem instances, with different algorithms performing well on different instances. This variation can be exploited by using *algorithm portfolios* [Huberman *et al.*, 1997; Gomes and Selman, 2001; Leyton-Brown *et al.*, 2003]. Such portfolios have proven successful across different areas of AI, including SAT solving [Xu *et al.*, 2008], AI planning [Helmert *et al.*, 2011] and Answer Set Programming [Gebser *et al.*, 2011]; see also the survey by Kotthoff [2014].

Algorithm portfolios are also useful in a second sense: as a context in which to study the contributions of various algorithms to the state of the art in solving a given problem. To see why this is important, consider the alternative: assessing algorithms based on their standalone performance. More formally (following the notation introduced by Fréchette *et al.* 2016), let X be a fixed set of instances of a given problem, let $A = \{1, \dots, \alpha\}$ be a portfolio of algorithms that solve this problem, let $\mathsf{perf}(A)$ be a measure of the performance achieved by A, and let $\mathsf{contr}(i, A)$ be the "contribution" of algorithm $i \in A$ to the performance of A. Then, evaluating each algorithm based solely on its standalone performance would be equivalent to setting: $\mathsf{contr}(i, A) := \mathsf{perf}(\{i\})$. The problem with this measure of contribution is that it fails to capture important qualitative differences in algorithm performance. To see why this is the case, consider instances x_1, x_2, x_3 and algorithms 1, 2, 3, 4, where 1 solves x_1 and x_2 within a given running time cutoff; 2 solves x_1; 3 solves x_2; and 4 solves x_3. Here, algorithm 1 has the best standalone performance; the other three algorithms are tied. However, by focusing solely on standalone performance, we miss the fact that algorithm 4 is special: it is the only one that solves x_3.

Xu *et al.* [2012] proposed that each algorithm i be evaluated in terms of its *marginal contribution* to portfolio A, or the improvement it achieves compared to the portfolio excluding i. Formally, they defined $\mathsf{contr}(i, A) := \mathsf{perf}(A) - \mathsf{perf}(A \setminus \{i\})$. This measure is able to capture the distinct performance of algorithm 4 in our example. It is the only algorithm with a positive marginal contribution. However, this measure, too, has a major shortcoming: algorithms with correlated strengths receive lower scores than they deserve. In our example, although algorithms 1, 2 and 3 are *collectively* important (they jointly solve two of the three instances, which cannot be solved in any other way), the marginal contribution of each of these algorithms is zero (given any two of 1, 2, 3, the third is unhelpful). Marginal contribution can fail to recognize significant performance differences, e.g., 1 solves twice as many instances as 2 and 3, yet they all have the same marginal contribution.

Fréchette *et al.* [2016] proposed a measure that addresses the aforementioned limitations. In particular, the authors model the portfolio as a coalitional game and calculate the Shapley value to determine the contribution of each algorithm. In a nutshell, the Shapley value for each algorithm is a weighted average of its marginal contributions over all subsets of the given portfolio A. In our example, the Shapley values (counting numbers of instances solved) for algorithms 1, 2, 3 and 4 are 1, 0.5, 0.5 and 1, respectively. There are good theoretical arguments for this measure and also intuitive reasons in its favour, as seen in our example, where it identifies algorithm 1 as being twice as important as 2 and 3; algorithms 2 and 3 receive identical scores; and 4 receives a higher score than 2 and 3, even though it, too, can only solve one instance, but is the only algorithm to solve that instance.

More formally, a coalitional game is defined by a pair (A, v), where $A = \{1, \ldots, \alpha\}$ is a set of α players and v is a characteristic function $2^A \rightarrow \mathbb{R}$ that maps each coalition $C \subseteq A$ to a real number, $v(C)$, called the *value* of coalition C, which represents the reward the players in C can achieve by working together. A game will often be denoted by just v instead of (A, v) whenever A is clear from the context. The coalition of all players $C = A$ is called the grand coalition. Denote the set of all games with players A as $\mathcal{V}(A)$.

A key concern for dividing a coalition's value amongst its players is fairness—a coalition's value should be distributed amongst the players in a manner reflecting the value that each player contributed. The canonical answer (or "solution concept") in this case is the Shapley value [Shapley, 1953]. It is based on the idea that, when players join a coalition in a fixed order, the contribution of any given player is taken as the increase in value that this player creates when joining the coalition, i.e., its "marginal contribution". The Shapley value is then defined as the average of such contributions over all possible joining orders. More formally, let $\Pi(A)$ denote the set of all permutations of A. For any $\pi \in \Pi(A)$, let C_i^π denote the coalition consisting of all predecessors of i in π. That is, $C_i^\pi = \{j \in A : \pi(j) < \pi(i)\}$, where $\pi(i)$ denotes the position of i in π. Then, noting that $v(C_i^\pi \cup \{i\}) - v(C_i^\pi)$ is the marginal contribution of player i to coalition C_i^π, the *Shapley value of player* $i \in A$ in game v is:

$$\phi_i(A, v) := \frac{1}{|A|!} \sum_{\pi \in \Pi(A)} v(C_i^\pi \cup \{i\}) - v(C_i^\pi). \quad (1)$$

Fréchette *et al.* [2016] modeled algorithm portfolios as coalitional games by assuming that each algorithm, or "solver", corresponds to a distinct player, and defining $v(C) = \mathsf{perf}(C)$. Then, the Shapley value for such a game becomes a "fair" measure of each algorithm's to the portfolio. However, the argument for scoring algorithms by their Shapley values breaks down when we wish to take into account the temporal order in which algorithms were invented, i.e., when we want to evaluate algorithms over time. To see why this is the case, let us revisit the example mentioned above. Imagine that algorithms 2 and 3 were published in 2015, 1 in 2016, and 4 in 2017. Then, by the time algorithm 1 was developed, we were already able to solve instances x_1 and x_2. From this perspective, algorithm 1 has not improved the state of the art, and the full credit for solving x_1 and x_2 should be attributed to algorithms 2 and 3. In contrast, even though 4 was published after the others, it advanced the state of the art by being the first to solve x_3.

This is the main idea behind the *temporal Shapley value* introduced in this paper: we specify temporal constraints between algorithms (reflecting the times at which they were introduced) and then average over all joining orders consistent with these constraints. In our example, and given the temporal constraints just described, we assign to algorithms 1, 2, 3 and 4 the scores 0, 1, 1, and 1, respectively. We axiomatize the new measure and prove that it maintains the Shapley value's desirable properties. We then show its usefulness in practice, analyzing the evolution of Quicksort pivoting strategies 1961–2009; entries in the SAT competitions 2007–2014; and entries in the MiniZinc competition for constraint solvers 2014–2016.

The temporal Shapley value is not limited to these case studies; indeed, it can be applied to any setting where performance values for each of a set of algorithms are available, for any performance measure. Computing it is efficient and takes a few seconds on a standard laptop for our case studies.

2 Coalitional Games in Temporal Settings

One of the assumptions underlying the Shapley value is that all joining orders are equally plausible. However, in the context of algorithms that were developed over time, this assumption does not hold. Instead, it would make more sense to define a partial order that constrains the joining orders such that an algorithm i is required to join before another j if the time of development of i precedes that of j (e.g., if i was introduced in an earlier iteration of an annual competition). This is particularly important in AI as new algorithms are often derived from older ones, e.g., by tweaking key features or adding new heuristics. Often, such *parent* and *child* algorithms are strongly correlated, with the child typically achieving a moderately better performance than the parent. If such a child-parent pair were evaluated together using the Shapley value, each of them would receive roughly half of the credit that either of them would have received if the other was absent. Moreover, the child would receive more credit, because adding the child to a coalition that contains the parent has a greater impact than vice versa. If we want to score algorithms only according to their overall usefulness, this assessment is reasonable: the child is indeed more effective than the parent. If, on the other hand, we want to apportion credit for beneficial ideas, it seems strange to neglect the fact that the child was derived from the parent. Importantly, this argument does not only hold for child-parent pairs (where one algorithm is an extension or improvement of the other), but for any pair of algorithms, as long as we are interested in attributing credit to the *first* algorithm that was able to solve certain problem instances.

We are thus interested in allowing only joining orders that reflect the time each algorithm was published. To formalize this idea, we define cooperative games with this restriction. Let $\mathcal{T}(A)$ denote a partition of A into q equivalence classes $T^1, \ldots, T^q$ (i.e., we assign the algorithms to q time periods, treating algorithms within each time period as incomparable). Let $Q : A \rightarrow \{1, \ldots, q\}$ denote an inverse function that maps from an algorithm to the index of its equivalence class. Define a *precedence relation* $\succ$ between algorithms, where $\forall i, j \in A$: $i \succ j$ iff $Q(i) \succ Q(j)$. Let $\mathcal{P}(A)$ denote the set of precedence relations that can be induced in this way (i.e., that are consistent with some partitioning of A and some ordering of the equivalence classes). Then, A and $\succ \in \mathcal{P}(A)$ induce a *strict partially ordered set* (or *poset*) of elements of A, which

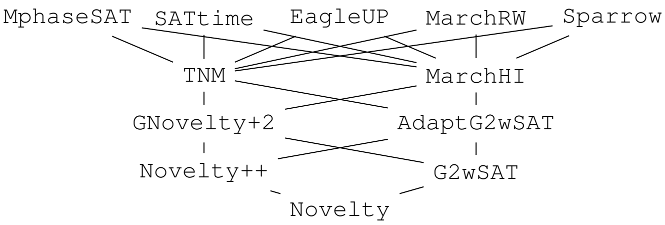

Figure 1: Precedence structure for the poset in our example. Nodes represent elements, edges represent the precedence relations between them; relations that can be deduced from transitivity are omitted.

we denote as $A^{\succ}$. A set of algorithms $C \in 2^A$ is *downward closed* if $\forall i \in C$: $i \succ j$ implies that $j \in C$. Given a poset $A^{\succ}$, let $\mathcal{C}(A^{\succ})$ denote the set of downward closed sets of elements of A. Finally, let $\Pi^{\succ}(A)$ be the set of permutations of A for which every prefix is an element of $\mathcal{C}(A^{\succ})$.

As an example, consider the following set of SAT solvers: $A = \{$Novelty, Novelty++, G2wSAT, GNovelty+2, AdaptG2wSAT, TNM, MarchHI, MphaseSAT, SATtime, EagleUP, MarchRW, Sparrow$\}$. These were introduced in the years 1997, 2005, 2005, 2007, 2007, 2009, 2009, 2011, 2011, 2011, 2011, and 2011, respectively. We define $\mathcal{T}(A)$ by creating an equivalence class corresponding to each of the five years, ordered by year, and assigning each solver to the appropriate class. Here, we have $j \succ i$ iff algorithm j was introduced after algorithm i. Figure 1 shows the precedence structure for the corresponding poset. Elements of $\mathcal{C}(A^{\succ})$ include any number of elements from any given equivalence class (rows in the diagram) and all elements of every prior (in the diagram, lower) equivalence class.

With this notation in place, we can now define *temporal coalitional games*.

Definition 1. *A* temporal coalitional game *is a triple* $(A, \succ, v^{\succ})$*, where A is the set of players,* $\succ \in \mathcal{P}(A)$ *is a precedence relation, and* $v^{\succ} : \mathcal{C}(A^{\succ}) \to \mathbb{R}$ *is the characteristic function defined on downward closed sets of players, with* $v^{\succ}(\emptyset) = 0$.

We will write $v^{\succ}$ instead of $(A, \succ, v^{\succ})$ when $A^{\succ}$ and $\succ$ are clear from context. Let $\mathcal{V}(A^{\succ})$ denote the set of all temporal coalitional games defined on $A^{\succ}$.

Temporal coalitional games are a special case of coalitional games under precedence constraints, where any precedence relations are admissible [Faigle and Kern, 1992]. It is useful to define the special case because it allows us to define and analyze a restricted version of the Shapley value that exploits the structure that holds here.

In this section, we define a solution concept $\phi^{\succ}$ that associates a temporal coalitional game $(A, \succ, v^{\succ})$ with a vector in $\mathbb{R}^{\alpha}$. We desire that $\phi^{\succ}$ should be uniquely determined by a set of axioms that is as close as possible to Shapley's original set (namely *Additivity*, *Efficiency*, *Null Player*, and *Symmetry*; see the online appendix, and that for a trivial precedence relation in which all algorithms are incomparable (e.g., were introduced in the same year), $\phi^{\succ}$ should reduce to the Shapley value.

We are able to use essentially the same Additivity and Efficiency axioms as for classical coalitional games.

Axiom 1′ (Additivity). *For any* $v^{\succ}, w^{\succ} \in \mathcal{V}(A^{\succ})$ *and* $C \in \mathcal{C}(A^{\succ})$*, let* $[v^{\succ} + w^{\succ}](C)$ *denote* $v^{\succ}(C) + w^{\succ}(C)$*. Then* $\phi^{\succ}(A, \succ, v^{\succ}) + \phi^{\succ}(A, \succ, w^{\succ}) = \phi^{\succ}(A, \succ, [v^{\succ} + w^{\succ}])$.

Axiom 2′ (Efficiency). *For any* $v^{\succ} \in \mathcal{V}(A^{\succ})$: $\sum_{i\in A} \phi_i^{\succ}(A, \succ, v^{\succ}) = v^{\succ}(A^{\succ})$.

The Null Player and Symmetry axioms need to be modified for the temporal setting. First, the Null Player axiom must account for the restrictions on the set of feasible coalitions.

Axiom 3′ (Null Player). *For any* $v^{\succ} \in \mathcal{V}(A^{\succ})$ *and* $i \in A$*, if* $v^{\succ}(C) - v^{\succ}(C \setminus \{i\}) = 0$ *for every* $C \in \mathcal{C}(A^{\succ})$ *such that* $i \in C$ *and* $C \setminus \{i\} \in \mathcal{C}(A^{\succ})$*, then* $\phi_i^{\succ}(v^{\succ}) = 0$.

The Symmetry axiom must consider that solvers belonging to different equivalence classes are asymmetric by definition; only solvers belonging to the same equivalence class can be required to be symmetric.

Axiom 4′ (Symmetry). *For any two players* $i, j \in A$ *for which* $Q(i) = Q(j) = p$, $\phi^{\succ}(A, \succ, f^p(v^{\succ})) = f^p(\phi^{\succ})(A, \succ, v^{\succ})$ *for every* $v^{\succ} \in \mathcal{V}(A^{\succ})$ *and every bijection* $f^p : A \to A$ *for which for any* $i \in A$, $i \notin T^p$ *implies that* $f^p(i) = i$.

We now turn to our main theoretical result: that there exists a unique $\phi^{\succ} : \mathcal{V}(A^{\succ}) \to \mathbb{R}^{\alpha}$ (which we will call *the temporal Shapley value*) satisfying our axioms, defined as:

$$\phi_i^{\succ}(A, \succ, v^{\succ}) = \frac{1}{|\Pi^{\succ}(A)|} \sum_{\pi \in \Pi^{\succ}(A)} \left(v^{\succ}(C_i^{\pi} \cup \{i\}) - v^{\succ}(C_i^{\pi})\right). \tag{2}$$

Lemma 1. *Equation 2 can be rewritten as*

$$\phi_i^{\succ}(A, \succ, v^{\succ}) = \sum_{C^p \subseteq T^p \setminus \{i\}} \frac{|C^p|!(|T^p \setminus C^p| - 1)!}{|T^p|!} MC_i(C^p), \tag{3}$$

where $p = Q(i)$ *and*

$$MC_i(C^p) = v^{\succ}\left(\bigcup_{k=1}^{p-1} T^k \cup C^p \cup \{i\}\right) - v^{\succ}\left(\bigcup_{k=1}^{p-1} T^k \cup C^p\right).$$

Proof. See the online appendix. □

This lemma makes it clear that the temporal Shapley value divides the credit amongst a set of algorithms introduced in the same year in exactly the same proportion as the classical Shapley value would.

Every temporal coalitional game has a characteristic function that is a linear combination of *simple characteristic functions (SCFs)*, where each SCF is identified with some downward-closed set C and asks whether the given coalition covers C. Formally, an SCF is defined based on a given $C \in \mathcal{C}(A^{\succ})$, $C \neq \emptyset$, and then evaluates to the following values given a coalition $D \in \mathcal{C}(A^{\succ})$:

$$\sigma_C(D) = \begin{cases} 1 & \text{if } D \supseteq C \\ 0 & \text{otherwise.} \end{cases}$$

We will now show that the simple games defined above form a basis of $\mathcal{V}(A^{\succ})$.

Lemma 2. *For every* $v^{\succ} \in \mathcal{V}(A^{\succ})$*, the game* $v^{\succ}$ *is equal to:*

$$\sum_{p=1}^{q} \sum_{\substack{C \subseteq T^p \\ C \neq \emptyset}} \left(\sum_{D \subseteq C} (-1)^{|C|-|D|} v^{\succ}\left(\bigcup_{r=1}^{p-1} T^r \cup D\right)\right) \sigma_{\bigcup_{r=1}^{p-1} T^r \cup C}.$$

Proof. See the online appendix. □

Theorem 1. *The temporal Shapley value is the unique solution concept satisfying Axioms 1′, 2′, 3′, and 4′.*

Proof. Consider a temporal game whose characteristic function is an SCF of the form $v^{\succ} = \sigma_E$ for some $E \in \mathcal{C}(A^{\succ})$, where $E = \bigcup_{k=1}^{p-1} T^k \cup E^p$. From the Efficiency and Null Player axioms, we know that $\phi_i^{\succ}(\sigma_E) = 0$ for all $i \in A \setminus E'$, and $\sum_{i \in E^p} \phi_i^{\succ}(\sigma_E) = 1$. Furthermore, from the Symmetry axiom, we conclude that:

$$\phi_i^{\succ}(\sigma_E) = \begin{cases} 0 & \text{if } i \in A \setminus E^p \\ \frac{1}{|E^p|} & \text{if } i \in E^p. \end{cases} \tag{4}$$

By the Additivity axiom, Equation 4 extends to the space of all games uniquely via Lemma 2, yielding Equation 3.

For the other direction of the proof, we have just seen that Equation 3 for $\phi^{\succ}$ satisfies Additivity, Efficiency, and Symmetry by construction. The Null Player axiom is also clearly satisfied—zero marginal contributions on the right hand side imply $\phi_i^{\succ}(v^{\succ}) = 0$. □

The temporal Shapley value can be classified as a special case of a general family of solution concepts called *quasivalues* [Monderer and Samet, 2002]. For a probability measure δ on $\Pi(A)$, a *quasivalue* q^δ assigns to $i \in A$ the payoff $q_i^\delta = \sum_{\pi \in \Pi(A)} \delta(\pi)\, (v(C_i^\pi \cup \{i\}) - v(C_i^\pi))$. The Shapley value is the quasivalue for the uniform distribution over joining orders, $\delta(\pi) = 1/(\alpha!)$ for all $\pi \in \Pi(A)$; see Equation 1. The temporal Shapley value is the quasivalue in which δ is uniform over joining orders restricted to the elements of $\Pi^{\succ}(A)$.

Fréchette *et al.* [2016] showed that, for characteristic functions that model popular competitions (corresponding to a well-known airport game), coalitional games can be represented succinctly using *marginal contribution networks* [Ieong and Shoham, 2005] which allows for the Shapley values to be computed in polynomial time. The same result holds in our setting of *temporal* coalitional games; see the online appendix.

3 Quicksort Over Time

Our first case study to illustrate the temporal Shapley value and the insights that can be gained from it involves the famous quicksort algorithm. In this context, we focus on the following strategies for picking the pivot that partitions the list to be sorted: (i) choosing the first element of the list from the original description of quicksort [Hoare, 1961]; (ii) choosing the middle element [Sedgewick, 1978]; (iii) choosing the median of a sample of three elements [Sedgewick, 1978]; (iv) choosing the median of a sample of nine elements [Bentley and McIlroy, 1993]; and (v) the dual-pivot scheme by Yaroslavskiy [2009] that is used in the standard quicksort implementation in Java. For all variants except the middle element and dual-pivot versions, we also consider randomized versions ("random" is the randomized version of "first"). We include insertion sort as a point of comparison that predates quicksort [Knuth, 1998].

We test the algorithms on lists of length 10 000, 50 000, 100 000, and 500 000. For each length, we follow Kushagra *et al.* [2014] and consider: (i) a list with all-equal elements; (ii) a list with already-sorted elements; (iii) a list with reverse-sorted elements; (iv) 1 000 lists with random permutations of all-unique elements; and (v) 1 000 lists with random permutations of repeated elements. Each sorting algorithm was run on each of these lists 100 times. The score of each run was the time the respective algorithm took to sort the respective list. [1]

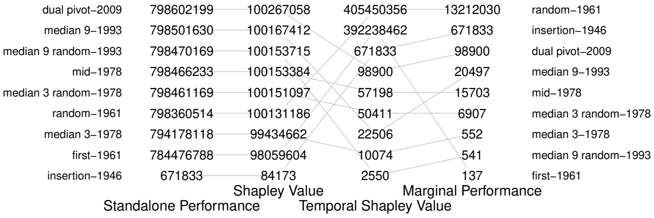

Figure 2: Comparison of standalone performance, Shapley value, temporal Shapley value, and temporal marginal contribution (marginal contribution to a temporally consistent portfolio) for insertion sort and different pivot strategies for quicksort.

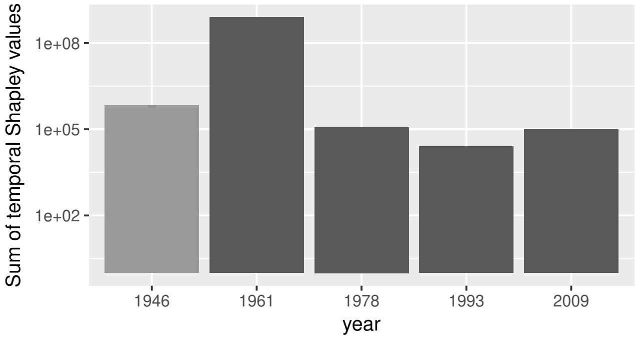

Figure 3: Year-over-year sum of temporal Shapley values for quicksort pivot strategies. Note the logarithmic y axis.

Figure 2 shows the ranking of the implementations. The most recent implementation has the best performance, while insertion sort performs worst. Interestingly, the ranking according to the Shapley value is exactly the same as for standalone performance. In contrast, according to the temporal marginal contribution, the original quicksort implementation that chooses a random element as pivot is ranked first, followed by insertion sort and the dual-pivot implementation.

These rankings do not tell the whole story—the dual-pivot implementation is used in practice, but it is unclear whether it provided the largest improvement over the previous state of the art. The temporal Shapley value shows that quicksort with a random pivot improved the most, starting quicksort's success story. The dual-pivot implementation ranks in the middle; it did not improve as much as the original 1961 implementations, but more than the incremental improvements thereof.

Figure 3 shows how much the state of the art has improved over the years. The initial quicksort implementations provide the largest improvement, while subsequent improvements are several orders of magnitude smaller and decreasing—changes become more incremental. The dual-pivot approach represents a major improvement, with a higher contribution than the previous ones. The temporal Shapley value clearly shows that this fundamentally different approach paid off.

[1] Code available at `https://git.io/vpEgM`.

4 SAT Solvers Over Time

We now use the temporal Shapley value to quantify the contributions to the state of the art made by the solvers participating in the SAT competition series [SAT Competitions Website, 2017]. Our experiments include solvers and problem instances from the 2007, 2009, 2011, 2013, and 2014 competitions. The competitions each consist of three *tracks* of problem instances: *random*, *crafted* and *application*. We excluded (i) solvers we were unable to obtain from the official SAT competition website or by contacting the authors; (ii) solvers that we could not build and run on our systems; and (iii) solvers that themselves use portfolio techniques.[2] In total, we considered 121 solvers: 38, 101 and 69 in the random, crafted and application tracks, respectively, with some solvers in multiple tracks.

For the random track, we considered a total of 1203 hard uniform k-SAT problem instances; for the crafted track, we considered 1029 instances manually designed to be challenging for SAT solvers; for the application track, we considered 1076 instances originating from applications of SAT to real world problems (e.g., software and hardware verification, cryptography, and planning). We considered both satisfiable and unsatisfiable instances. As in the 2014 SAT Competition, we gave each run 14 GB of memory and 5000 CPU sec.

The SAT competition ranks solvers by the number of instances they solve, breaking ties according to runtime. We adopt the *single scoring* function introduced by Fréchette *et al.* [2016], which models this approach using a single real value. More specifically, we define the score of an algorithm $i \in A$ on an instance $x \in X$ as:

$$\mathsf{score}_x(i) = \begin{cases} 0 & x \text{ not solved by } i \\ 1 + \frac{c-t}{|X|\cdot c\cdot |A|+1} & \text{otherwise,} \end{cases} \tag{5}$$

where c is the maximum CPU time allowed for solving an instance, and t is the CPU time required for solver i to solve instance x. The score of a set of solvers $A' \subseteq A$ given instance x is then:

$$\mathsf{score}_x(A') = \max_{i \in A'} \mathsf{score}_x(i). \tag{6}$$

The performance of A' on a set of instances is the sum of the scores of A' on all those instances.

Figure 4 shows the results for the random track (results for the remaining tracks were qualitatively similar, and were omitted due to limited space). It becomes immediately obvious that the temporal Shapley value gives a much more accurate picture of the relative importance of the contributions of the algorithms. In particular, while all 2007 solvers rank quite lowly in terms of Shapley value, some of them rank very highly in terms of the temporal Shapley value. We observed the most radical changes in rank for the 2007 solvers `March-KS` and `KCNFS`. These no longer contribute much to the state of the art, but had a large impact in 2007, when they first participated in the competition. The 2014 solver `dimetheus`, on the other hand, ranked first in terms of Shapley value, but only ninth according to the temporal version. This shows that, although `dimetheus` provides great performance, a significant part of its performance advantage over most solvers comes on instances that could also be solved by some of the strongest early solvers.

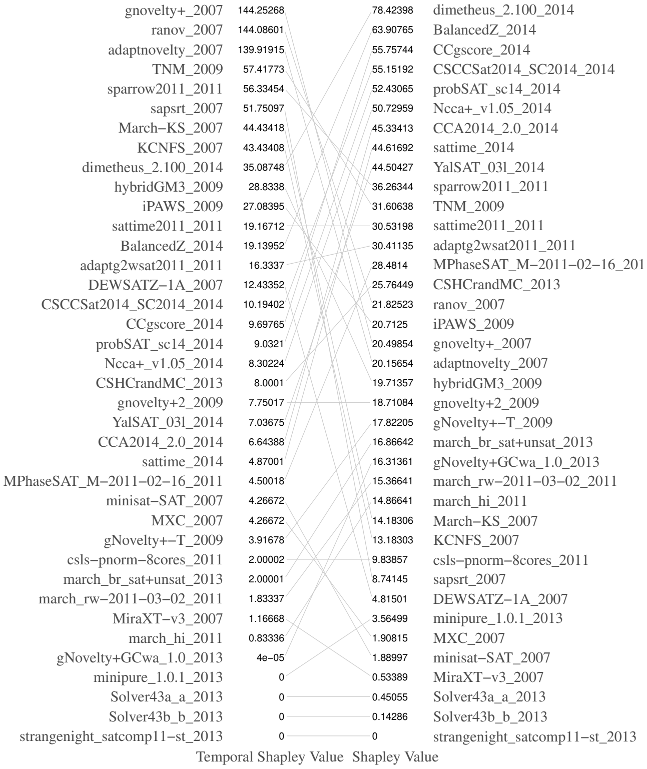

Figure 4: Comparison of classic and temporal Shapley values for the SAT competition solvers 2007-2014, random track.

Figure 5 shows the year-over-year performance increase of the state of the art, quantified by the sum of temporal Shapley values. The improvement over the previous year was largest in 2009, and decreased to almost zero in 2013, which appears to have been a bad year for innovation in the random track of the SAT competition. In 2014 there was a substantial improvement again. This was mostly because of `dimetheus`, which made a very strong contribution in terms of temporal Shapley value.

5 Constraint Solvers Over Time

We now consider the MiniZinc challenge [Stuckey *et al.*, 2014]. We conducted experiments on the solvers from the 2014 (16 solvers), 2015 (20 solvers), and 2016 (25 solvers) challenges. We excluded the solvers `SunnyCP` and `MZNGurobi`, because both required access to a custom license server set up for the competition for the commercial `Gurobi` solver. We used solvers from the 2014, 2015, and 2016 challenges, because virtual machines with the runtime environment, compilation options, and call parameters were not available for other years. We used the problem instances from the 2013, 2014, 2015, and 2016 challenges (100 instances each). The challenges were divided into *finite domain*, *free*, and *parallel search* tracks. The cutoff time for all solvers on all instances was 1200 seconds.

The MiniZinc challenge uses a scoring function based

[2] We were unable to build 2 solvers from 2014, 2 from 2013, 8 from 2011, 5 from 2009, and 5 from 2007. We were also unable to run one solver from 2014, one from 2013, and one from 2009.

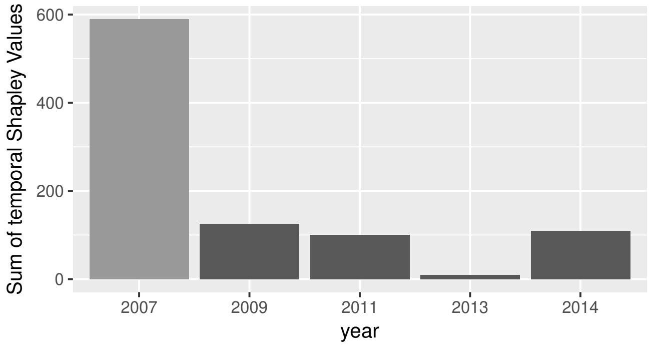

Figure 5: Year-over-year change of the sum of temporal Shapley values for the SAT competition 2007–2014, random track.

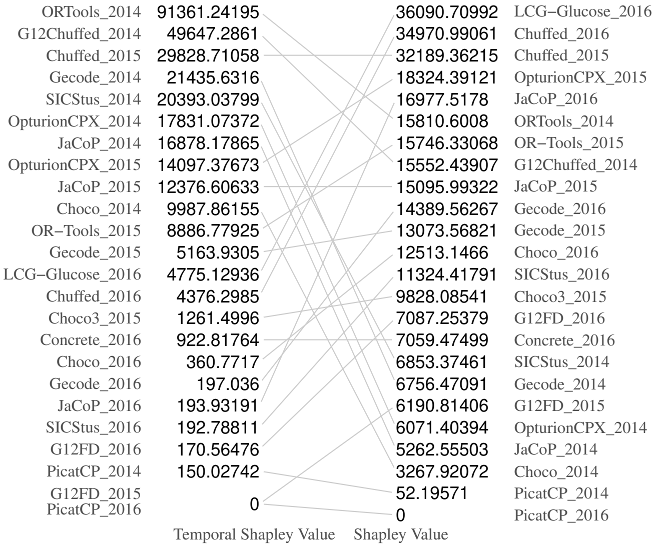

Figure 6: Comparison of classic and temporal Shapley values for the MiniZinc challenge, finite domain track.

on Borda voting. Each instance “ranks” solvers, which are awarded points proportional to the number of solvers they beat. This scoring function takes relationships between solvers into account, which is undesirable for our purposes. We therefore scored runs by running time.

Figure 6 shows Shapley values and temporal Shapley values for the finite domain track. The results for the other tracks were qualitatively similar (not shown due to limited space). We observed similar results as for the SAT competition—early solvers made greater contributions in terms of the temporal Shapley value than in terms of Shapley value, and most (but not all) of the highest-ranked solvers came from the first year of the challenge. For example, the highest-ranked solver in terms of the temporal Shapley value, `ORTools`, was entered in the first year but ranked only sixth in terms of Shapley value. `LCG-Glucose`, entered in 2016, ranked first in terms of Shapley value, but only 13th according to the temporal Shapley value.

Figure 7 shows the year-over-year change of the improvement. As for the SAT competition, performance increases diminished over time, with the lowest improvement in the most recent year. The other tracks of the MiniZinc challenge were similar, although 2016 showed a slightly greater performance increase than 2015 in the parallel search track.

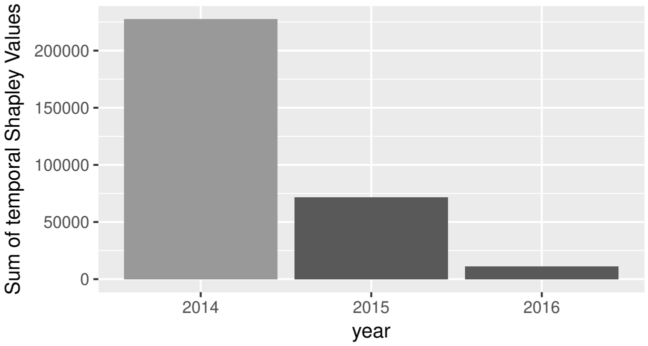

Figure 7: Year-over-year change of the sum of temporal Shapley values for the MiniZinc challenge solvers 2014–2016, finite domain track.

6 Conclusions

We introduced a principled approach to evaluating the contributions that individual solvers make to the state of the art while taking time into account: the temporal Shapley value. Our work complements previous work that quantified the contributions of algorithms with the (classic) Shapley value, which is appropriate for use in cases where all algorithms are on the same footing, but can create the misleading impression that older algorithms contributed little to the state of the art, even though they form the basis for newer algorithms.

We laid firm theoretical foundations for our work, extending coalitional game theory to a setting in which coalitions must form in a way consistent with temporal constraints. We defined an analogue of the classic Shapley value for the temporal setting, showing axiomatically that it preserves the beneficial properties of the classic Shapley value.

We then performed an experimental analysis of the well-known quicksort algorithm and solvers participating in international competition in two prominent areas of AI, namely SAT solving and Constraint Programming, to demonstrate the impact of our framework. These examples illustrated that the temporal Shapley value reflects the impact that an algorithm had on the state of the art more accurately than the Shapley value—early algorithms get the credit they deserve. We also performed an aggregate year-over-year analysis, which showed that the rate of progress in both AI fields has slowed in recent years.

Acknowledgements

We thank Andreas Schutt for kindly providing the virtual machines of the MiniZinc challenge solvers and helping to set up the experiments. 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. Tomasz P. Michalak was supported by by the Polish National Science Centre grant 2015/19/D/ST6/03113.

References

[Aadithya *et al.*, 2011] K.V. Aadithya, T.P. Michalak, and N.R. Jennings. Representation of coalitional games with algebraic decision diagrams. In *10th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1121–1122, 2011.

[Bentley and McIlroy, 1993] Jon L. Bentley and M. Douglas McIlroy. Engineering a sort function. *Software: Practice and Experience*, 23(11):1249–1265, 1993.

[Carlsson *et al.*, 1997] Mats Carlsson, Greger Ottosson, and Björn Carlson. An open-ended finite domain constraint solver. In *Programming Languages: Implementations, Logics, and Programs, 9th International Symposium*, pages 191–206, 1997.

[Chalkiadakis *et al.*, 2011] G. Chalkiadakis, E. Elkind, and M. Wooldridge. *Computational Aspects of Cooperative Game Theory*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

[Elkind *et al.*, 2009] E. Elkind, L.A. Goldberg, P.W. Goldberg, and M. Wooldridge. A tractable and expressive class of marginal contribution nets and its applications. *Mathematical Logic Quarterly*, 55(4):362–376, 2009.

[Faigle and Kern, 1992] Ulrich Faigle and Walter Kern. The Shapley value for cooperative games under precedence constraints. *International Journal of Game Theory*, 21(3):249–266, 1992.

[Fréchette *et al.*, 2016] Alexandre Fréchette, Lars Kotthoff, Talal Rahwan, Holger H. Hoos, Kevin Leyton-Brown, and Tomasz P. Michalak. Using the Shapley value to analyze algorithm portfolios. In *30th AAAI Conference on Artificial Intelligence*, pages 3397–3403, February 2016.

[Gebser *et al.*, 2011] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Torsten Schaub, Marius Thomas Schneider, and Stefan Ziller. A portfolio solver for answer set programming: preliminary report. In *11th International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 352–357. Springer, 2011.

[Gomes and Selman, 2001] Carla P. Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1–2):43–62, 2001.

[Helmert *et al.*, 2011] Malte Helmert, Gabriele Röger, and Erez Karpas. Fast downward stone soup: A baseline for building planner portfolios. In *ICAPS-2011 Workshop on Planning and Learning (PAL)*, pages 28–35, 2011.

[Hoare, 1961] C. A. R. Hoare. Algorithm 63: Partition. *Commun. ACM*, 4(7):321, July 1961.

[Huberman *et al.*, 1997] Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An economics approach to hard computational problems. *Science*, 275:51–54, January 3 1997.

[Ieong and Shoham, 2005] S. Ieong and Y. Shoham. Marginal contribution nets: a compact representation scheme for coalitional games. In *6th ACM Conference on Electronic Commerce*, pages 193–202, 2005.

[Knuth, 1998] Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., 1998.

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

[Kushagra *et al.*, 2014] Shrinu Kushagra, Alejandro López-Ortiz, J. Ian Munro, and Aurick Qiao. Multi-pivot Quicksort: Theory and Experiments. In *Meeting on Algorithm Engineering & Expermiments*, pages 47–60. Society for Industrial and Applied Mathematics, 2014.

[Leyton-Brown *et al.*, 2003] Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, James McFadden, and Yoav Shoham. A portfolio approach to algorithm selection. In *18th International Joint Conference on Artificial Intelligence*, pages 1542–1543, 2003.

[Michalak *et al.*, 2010] T.P. Michalak, D. Marciniak, M. Samotulski, T. Rahwan, P. McBurney, M. Wooldridge, and N.R. Jennings. A logic-based representation for coalitional games with externalities. In *9th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 125–132, 2010.

[Monderer and Samet, 2002] Dov Monderer and Dov Samet. Variations on the Shapley value. In *Handbook of Game Theory with Economic Applications*, volume 3, chapter 54, pages 2055–2076. Elsevier, 1st edition, 2002.

[SAT Competitions Website, 2017] SAT Competitions Website. `http://www.satcompetition.org`, 2017.

[Sedgewick, 1978] Robert Sedgewick. Implementing Quicksort Programs. *Commun. ACM*, 21(10):847–857, October 1978.

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

[Solan *et al.*, 2013] Eilon Solan, Shmuel Zamir, and Michael Maschler. *Game Theory*. Cambridge University Press, 2013.

[Stuckey *et al.*, 2014] Peter J. Stuckey, Thibaut Feydy, Andreas Schutt, Guido Tack, and Julien Fischer. The MiniZinc Challenge 2008-2013. *AI Magazine*, 35(2):55–60, 2014.

[Xu *et al.*, 2008] 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.

[Xu *et al.*, 2012] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Evaluating component solver contributions to portfolio-based algorithm selectors. In *15th International Conference on Theory and Applications of Satisfiability Testing*, pages 228–241. Springer-Verlag, 2012.

[Yaroslavskiy, 2009] Vladimir Yaroslavskiy. Dual-Pivot Quicksort algorithm. `http://codeblab.com/wp-content/uploads/2009/09/DualPivotQuicksort.pdf`, 2009.