Revisiting Action Failures Through the Lens of Cooperative Games

Sarath Sreedharan[1], **David Smith**[2]

[1]Computer Science Department, Colorado State University, Fort Collins, USA
[2]PS Research, Los Altos Hills, CA, USA
ssreedh3@colostate.edu, david.smith@psresearch.xyz

Abstract

This paper will revisit the question of explaining action or plan failures in the context of numeric planning problems. As we will see, the existing methods used in classical planning for explaining plan failures fall short in this setting, and we need a novel paradigm to handle this question. To model such explanation methods, we turn to cooperative games as a framework to capture such explanation generation problems. Then we look at two possible methods, namely, Shapley values and Banzhaf values, to identify the role played by each action in the resulting failure. While such methods have been previously used in explaining machine learning models, we see that their deployment in planning reveals novel challenges. As we will see, we will need to modify how these values are calculated for the planning setting. Finally, through a running example, we will see how explanations may be calculated using each of these methods, and how they may result in different explanations. This points to how more work needs to be done to evaluate which of these methods may be better suited for planning settings.

Introduction

Explaining plan or action failures is crucial when analyzing plans provided by humans or other agents. When an action condition fails, it is easy to identify the failing condition, but this doesn't explain why the condition is not satisfied. A more informative and helpful explanation involves identifying previous action(s) that resulted in this condition being violated[1]. For classical planning, this corresponds to identifying the action that deleted the fact subsequent to the last point at which the fact may have been true. Now, one could disagree on the finer points, like whether a better explanation might be the first action that deletes the fact or the deleting action that is closest to the failing action. Regardless of the specific action identified as the cause, identification of the explanation remains both conceptually and computationally relatively simple. After all, the worst-case complexity of identifying this explanation is polynomial in the length of the plan.

However, as one leaves the realm of classical planning and enters that of numeric planning, the problem of explaining action failures takes on new dimensions. To illustrate this, consider the following example, where we have a rover moving to a crater, taking pictures, collecting soil samples, and communicating them to the lander. Now, the exact plan is given as follows:

$$\pi = \langle \mathsf{move\_to\_crater}, \mathsf{scan\_for\_rocks}, \mathsf{take\_picture\_of\_crater}, \mathsf{pickup\_soil\_sample}, \mathsf{communicate\_image} \rangle$$

Assume that the rover starts with a full battery. The first move action (referred henceforth as Mv) uses up 30% of the battery level, while the scan action (Scn) uses up 5%, and the next two actions use up 20% each (Tk and Pup). Finally, the communicate_image action (Comm) requires the battery level to be at least 52%. Comm would therefore fail, as the battery level would only be 25% at the start of the action. Which prior action was responsible for this failure? A quick scan through the plan shows that it's difficult to pinpoint a single cause for failure because all of the actions preceding Comm reduce the battery level. Simply citing all of the actions that influence the battery level is also not very helpful, as it could be a very long list for a larger plan. While one could point to Mv as a potential reason, due to the high battery usage, that action on its own doesn't reduce the battery to the degree that Comm can no longer be performed. One could instead argue that Tk is the responsible action since it first reduces the battery level below the required level of 52%. However, removing that action doesn't necessarily fix the failure, as Pup would still lead to the failure. The only thing we could say with certainty is that Scn had very little role to play in the failure , after all, removing it or adding it back makes no difference in the context of the other actions. In this case, what we need is a mechanism that will allow us to assign a degree of blame to each preceding action and use this to understand the role each action plays in the resulting failure.

This brings us to the focus of this paper: looking at the problem of explaining plan failures through the lens of cooperative games (Driessen 2013). Here, each action becomes a player working towards the objective of causing the target precondition to be violated. Any coalition of players that results in such a failure receives a value of 1 and 0 otherwise. We will see how methods for attributing player contributions could be used here to identify the role played by each action

[1]One could use this paradigm to capture cases where a fact was never true to start with, by creating an action a_0 that would stand-in for setting the initial state.

in the resulting failure. While mapping explanations to outcome attributions in cooperative games has been studied in the context of machine learning models (Sundararajan and Najmi 2020), to the best of our knowledge, this presents the first attempt at mapping action failure explanation to a cooperative game. We will see how the applications of these methods in planning introduce novel challenges not present in traditional one-shot machine learning prediction settings. The Shapley value (Winter 2002) is the most popular value attribution method used in explainable prediction settings, but we will also consider the Banzhaf value (Lehrer 1988). We believe these two different attribution methods capture two primary ways in which one could think of plans, namely, as a totally ordered action sequence or as a partially ordered set of actions. These will be described in greater detail in the next section.

Background

In this section, we set up some of the foundational notations we will need to discuss how one could map plan failure to a generalized cooperative game. To start with, we will represent our planning problem or model using the tuple $\mathcal{M} = \langle F, A, I, G \rangle$, where F represents the set of fluents, A the actions, I the initial state, and G the goal. Since we are interested in numeric planning problems, our fluent set will be represented as $F = \langle F_p, F_n \rangle$, where F_p captures the propositional fluents and F_n the numeric ones. Our initial state, goal description, and action definitions will be defined over both the propositional and numeric fluents. In particular, for action definitions, the preconditions of the actions defined over numeric fluents will either take the form of $(f \diamond i)$ or $(f \diamond g)$, where $f, g \in F_n$, $i \in \mathbb{R}$ is a real constant, and $\diamond$ is a relational operator in $\{<, \leq, =, \geq, >, \neq\}$. The effects of the actions defined over numeric fluents are given as $(f \leftarrow \text{expr})$, where $f \in F_n$, and expr is an arithmetic operation defined over numeric fluents and real-valued constants. Abusing the notation, we will denote the effect of executing action a in state s as $a(s)$ if the preconditions of action a are satisfied in s. Similarly, we will denote the value of a fluent f in state s as $s(f)$. A solution to a planning problem is a plan π, whose execution in the initial state I, results in a state that satisfies the goal, namely $\pi(I) \models G$. In addition to plans, we will also leverage the notion of an executable action sequence, which is an action sequence whose execution in the initial state will result in a valid state. In other words, there exists no action in the sequence whose preconditions aren't satisfied by preceding actions.

Another core concept we will use in this paper is alternative representations of plans or plan prefixes. Firstly, we will consider the totally ordered representation, where a plan is represented by an action sequence of the form $\pi = \langle a_1, ..., a_k \rangle$. We will also consider a partially ordered representation, where a plan is represented by a tuple of the form $\pi_\prec = (\mathbb{A}, \prec)$. Here, $\mathbb{A}$ is a set of unique steps of the plan. Here, each step of the plan maps over to an action in the set A, but multiple steps may refer to the same action. Next, $\prec$ is a precedence ordering between elements in $\mathbb{A}$. $\pi_\prec$ is considered a valid plan **if there exists** a total ordering for elements in $\mathbb{A}$, denoted as π', such that π' satisfies $\prec$ and $\pi'(I) \models G$. We will extend the notion to also support executable action sequences. A partially ordered set of actions $\tilde{\pi}_\prec = \langle \tilde{\mathbb{A}}, \prec \rangle$ is said to be executable **if there exists** a total ordering for the elements in $\mathbb{A}$ that is executable.

Cooperative Games. In a cooperative game (Driessen 2013), a set of players works together towards a common objective or payoff. Within the game, a set of players is expected to form groups or coalitions, where the players within the coalitions are expected to work together towards the shared objective. The games also include a characteristic function or a value function that determines the total payoff achieved by the coalition. Here, the value function assigns a numeric value to the coalition, with the assumption that a higher value means a higher pay-off. Since in this game all the players are cooperating, a common question one could ask is, what is the share of the total pay-off each player deserves? Such allocations are usually done through means like Shapley or Banzhaf values (Chalkiadakis, Elkind, and Wooldridge 2011).

More formally, we define the cooperative game $\Gamma = \langle \mathcal{N}, \mathcal{V} \rangle$, where $\mathcal{N}$ is the set of all players, such that $|\mathcal{N}| = n$ and $\mathcal{V}$ is the value function that assigns a value to coalitions formed by a set of players. Here, each coalition C corresponds to a subset of players. Here, a coalition containing all players, i.e., $C = \mathcal{N}$, is referred to as the grand coalition.

An important subcategory of cooperative games is that of simple games or *win-or-lose* games (Maschler, Zamir, and Solan 2020). Two important characteristics of such games are the fact that either a given coalition can win (getting a pay-off of 1) or lose (getting a payoff of 0). Additionally, these games are monotonic. Here, monotonicity refers to the fact that adding more players to a winning team or coalition cannot turn it into a losing one. Or more formally, if $\mathcal{V}(C) = 1$, then for any $C' \supseteq C$, $\mathcal{V}(C') = 1$. These capture many natural cases where bringing more members into the coalition won't make it weaker. A related but distinct concept is that of zero-sum games (Von Neumann and Morgenstern 1947). Zero-sum games usually represent competitive game settings, where one player gets all the pay-off. For example, in many cases, the possible utility of the players in such games might be represented as '0' or '1', but there is no notion of coalitions, and at the end, only one of those players might receive the non-zero utility (as such it's also referred to as winner-takes-all games). On the other hand, for a simple game, while the coalition may get a pay-off of 1, this pay-off is distributed across each of the players. The amount they receive is proportional to the role played by the player in reaching the outcome. In this paper, we will look at cooperative games, which have a binary pay-off but aren't monotonic (for reasons we will see later).

As discussed, one of the popular ways of distributing the total payoff across players is through Shapley values (Sundararajan and Najmi 2020), denoted as ϕ_i, which is given by

$$\phi_i(\mathcal{V}) = \sum_{C \subseteq \mathcal{N} \setminus i} \frac{|C|!(n - |C| - 1)!}{n!} * (\mathcal{V}(C \cup \{i\}) - \mathcal{V}(C)) \tag{1}$$

Here $\mathcal{V}(C \cup i) - \mathcal{V}(C)$ is the marginal contribution of adding player i to the coalition C (i.e., does the total pay-off increase when we add this player to the team). The term $|C|!$ captures the ways to order the members of the current coalition, and $(n - |C| - 1)!$ represents all the ways the players excluding i and the ones in C could be arranged. This means each term in the sum is considering the fraction of permutations, where the player i follows some ordering over the current coalition, and is followed by all the remaining players. As such, an important aspect of the Shapley value is that it considers all possible permutations in which the overall team can be formed and averages the contributions across all possible permutations. One of the important properties that Shapley values have is the efficiency property, namely that it should distribute the values across the different players.

$$\sum_{i} \phi_i(\mathcal{V}) = \mathcal{V}(\mathcal{N})$$

An alternative to the Shapley value is the Banzhaf value. While the Banzhaf value has historically been used primarily in simple games, it provides a formulation that ignores the individual permutations possible for coalitions. The Banzhaf value for a player i is given by:

$$\beta_i(\mathcal{V}) = \frac{1}{2^{n-1}} * \sum_{C \subseteq \mathcal{N} \setminus i} (\mathcal{V}(C \cup \{i\}) - \mathcal{V}(C)) \quad (2)$$

The main difference is that the Banzhaf value doesn't consider how many permutations can be formed from each coalition, and hence isn't multiplying the marginal contribution by the number of permutations. As we will see, for cases where not all permutations are possible, this will result in extremely divergent results on the ordering between Banzhaf and Shapley value-based ordering of players. Additionally, the Banzhaf value doesn't try to satisfy the efficiency axiom since the individual Banzhaf values do not need to add up to the total pay-off. Instead, this formulation only requires that the values add up to the power of the game, where power $\mathcal{P}(\mathcal{N}, \mathcal{V})$ is proportional to the sum of all marginal contributions, i.e.,

$$\mathcal{P}(\mathcal{N}, \mathcal{V}) = \frac{1}{2^{n-1}} * \sum_{i} \sum_{C \subseteq \mathcal{N} \setminus i} (\mathcal{V}(C \cup \{i\}) - \mathcal{V}(C))$$

With these basic concepts in place, we can move on to the main proposal of our work.

Mapping Action Failures to Generalized Cooperative Games

The primary contribution of this paper is to show how one might cast the explanation of an action condition failure as a generalized cooperative game. To start with, let's consider a planning model $\mathcal{M} = \langle F, A, I, G \rangle$, and a sequence of actions $\pi = \langle a_1, ..., a_k \rangle$, such that the execution of π results in the failure of action a_k, i.e., if $\hat{\pi} = \langle a_1, ..., a_{k-1} \rangle$, then $\hat{\pi}(I) \not\models \text{pre}(a_k)$, where $\text{pre}(a_k)$ is the preconditions of the action a_k. To be more specific, let the failing condition be $f_k \diamond_k M$, where $f_k \in F_N$, $\diamond_k$ is a relational operator, and M is a real-valued constant.[2] Now, we define the corresponding game as $\Gamma^{\pi} = \langle \hat{\mathcal{N}}, \hat{\mathcal{V}} \rangle$, where $\hat{\mathcal{N}}$ contains a player for each index in $\hat{\pi}$. Here, a coalition could be thought of as a partially ordered set of actions with no ordering constraints. The objective of the game is to bring about a state where the target conditions $f_k \diamond_k M$ fail. As such, our value function returns 1 for coalitions or action sets, which could lead to the failure of the condition. More formally, the value for a coalition C is defined as

$$\hat{\mathcal{V}}(C) = \begin{cases} 1 \text{ if } C \not\models f_k \diamond_k M \\ 0 \text{ otherwise} \end{cases}$$

The players in the game consist of actions from the executable prefix $\hat{\pi}$ before the failing action a_k. As a result, we are looking for an executable partially ordered set of actions from that sequence that leads to a state where the condition $f_k \diamond_k M$ doesn't hold. More specifically, $C \not\models f_k$ stands for the fact that there exists an executable linearization π' for C such that π' is a valid action sequence whose execution results in a state where the condition $f_k \diamond_k M$ does not hold. Now, let us look at the individual values. Note that while our game formulation has a 1 or 0 value, it isn't a simple game since the monotonicity property is violated. Specifically, in certain cases, adding a new action to the coalition could make a winning coalition (one that violates the condition) move to a losing one (one that satisfies the condition). For example, let's consider a variation of our rover example with a recharge (RC) action that sets the battery back to 100%. Here, a coalition consisting of actions $\{\mathsf{Mv}, \mathsf{Tk}\}$ is a winning coalition because it violates the condition that the battery should be above 52%. Now, if we were to add the action RC to the coalition, it turns into a losing coalition.

Shapley Values As one attempts to look at Shapley values in the context of planning, the first fact that stands out is that not every permutation of any given coalition is feasible for the given planning model M. This means that to find the Shapley value, we need to find all possible feasible permutations of the grand coalition, i.e., the ones containing all actions denoted by $\Pi^{\mathcal{N}}$, and all permutations of the actions that start with the elements in a given coalition C before a player i, denoted by Π_i^C. As such, the new Shapley value for player i becomes

$$\phi_i(\hat{\mathcal{V}}) = \sum_{C \subseteq \hat{\mathcal{N}} \setminus i} \frac{|\Pi_i^C|}{|\Pi^N|} * (\hat{\mathcal{V}}(C \cup \{i\}) - \hat{\mathcal{V}}(C)) \quad (3)$$

A proof that our updated method still satisfies this property is given in the appendix. It is worth noting that this notion that certain permutations of the players aren't a common consideration in cooperative games, and to the best of our knowledge, has never been considered in the context of the use of Shapley values in explanations for machine learning. However, this is a very natural notion within the context of

[2]Please note that our focus on real-valued constants is made purely to simplify the discussion. We can easily support cases where the precondition is defined over other numeric fluents or even more complex expressions.

planning problems, where we are always talking about interdependencies between the actions.

Revisiting our example, suppose that there is a precondition on Mv, such that it cannot be executed after any of the following actions. Also suppose that once the rover reaches the crater, it can perform the other three actions in any order. Since Shapley values consider the permutations of all actions, the total number of sequences is given as $|\Pi^N| = 6$. Specifically, the possible permutations are as follows:

$$[\langle \mathsf{Mv}, \mathsf{Scn}, \mathsf{Tk}, \mathsf{Pup}\rangle, \langle \mathsf{Mv}, \mathsf{Scn}, \mathsf{Pup}, \mathsf{Tk}\rangle,$$
$$\langle \mathsf{Mv}, \mathsf{Tk}, \mathsf{Scn}, \mathsf{Pup}\rangle, \langle \mathsf{Mv}, \mathsf{Tk}, \mathsf{Pup}, \mathsf{Scn}\rangle,$$
$$\langle \mathsf{Mv}, \mathsf{Pup}, \mathsf{Tk}, \mathsf{Scn}\rangle, \langle \mathsf{Mv}, \mathsf{Pup}, \mathsf{Scn}, \mathsf{Tk}\rangle]$$

Here, there are 16 possible coalitions that correspond to all possible subsets of $\{\mathsf{Mv}, \mathsf{Scn}, \mathsf{Tk}, \mathsf{Pup}\}$. Let's assume that all of them are executable, in the sense that there exists a linearization for the corresponding partially ordered set that is executable. Now, the only winning coalitions, i.e., the ones that return 1 are $\{\mathsf{Mv}, \mathsf{Scn}, \mathsf{Tk}, \mathsf{Pup}\}$, $\{\mathsf{Mv}, \mathsf{Scn}, \mathsf{Tk}\}$, $\{\mathsf{Mv}, \mathsf{Scn}, \mathsf{Pup}\}$, , $\{\mathsf{Mv}, \mathsf{Tk}, \mathsf{Pup}\}$, $\{\mathsf{Mv}, \mathsf{Tk}\}$, and $\{\mathsf{Mv}, \mathsf{Pup}\}$. Note that in the Shapley value, for each coalition, we are multiplying the marginal contribution of the player by the number of permutations where the players from that coalition precedes the current player. For any player i, the number of permutations where C precedes i ($|\Pi_i^C|$) has a non-zero value only for coalitions that correspond to the empty set or ones containing Mv. This is because every valid permutation starts with Mv, which can be said to precede the empty set by default. For player Mv, since we are iterating over coalitions formed from subsets of $\hat{\mathcal{N}}, \setminus\{\mathsf{Mv}\}$, it turns all coalitions not corresponding to the empty set into zero. Thus, the Shapley value for Mv becomes:

$$\phi_{\mathsf{Mv}}(\hat{\mathcal{V}}) = \frac{|1}{|\Pi^N|} * (\hat{\mathcal{V}}(\emptyset \cup \{\mathsf{Mv}\}) - \hat{\mathcal{V}}(\emptyset)) = 0$$

The above value is zero, because neither the empty set of players nor Mv on its own violates the condition. This means according to Shapley value Mv gets no credit for violating the condition. It never turns a losing coalition into a winning one, over the valid action sequences. Suppose we were to calculate the Shapley values for the rest of the players in a similar fashion, we will see:

$$\phi_{\mathsf{Scn}}(\hat{\mathcal{V}}) = 0$$
$$\phi_{\mathsf{Tk}}(\hat{\mathcal{V}}) = 1/2$$
$$\phi_{\mathsf{Pup}}(\hat{\mathcal{V}}) = 1/2$$

Again, Scn gets zero value because it never turns a losing coalition into a winning one, and the rest of the value is distributed equally among the other players.

Banzhaf Values For the Banzhaf value, we again need to update the calculation. Here, rather than considering all possible subsets, we only consider the subsets that correspond to an executable partially ordered set of action that excludes the current action (i.e., for which there exists a valid total ordered plan). Let's denote this as $\Pi_{\prec}^{-i}$. The Banzhaf value calculation becomes:

$$\beta_i(\hat{\mathcal{V}}) = \frac{1}{|\Pi_{\prec}^{-i}|} * \sum_{C \subseteq \Pi_{\prec}} (\hat{\mathcal{V}}(C \cup \{i\}) - \mathcal{V}(C))$$

The proof that this still satisfies the property that it adds up to the power can be trivially seen to be true.

Revisiting the example, we see that the Banzhaf value gives a surprisingly different result. The main difference is that Banzhaf iterates over the winning coalitions, which as we pointed out earlier are $\{\mathsf{Mv}, \mathsf{Scn}, \mathsf{Tk}, \mathsf{Pup}\}$, $\{\mathsf{Mv}, \mathsf{Scn}, \mathsf{Tk}\}$, $\{\mathsf{Mv}, \mathsf{Scn}, \mathsf{Pup}\}$, , $\{\mathsf{Mv}, \mathsf{Tk}, \mathsf{Pup}\}$, $\{\mathsf{Mv}, \mathsf{Tk}\}$, and $\{\mathsf{Mv}, \mathsf{Pup}\}$. As we discussed, all possible subsets correspond to an executable partially ordered set of actions, thus $|\Pi_{\prec}^{-i}| = 8$ for every player i. Now, calculating the Banzhaf value, we see the individual values to be:

$$\beta_{\mathsf{Mv}}(\hat{\mathcal{V}}) = 3/4$$
$$\beta_{\mathsf{Scn}}(\hat{\mathcal{V}}) = 0$$
$$\beta_{\mathsf{Tk}}(\hat{\mathcal{V}}) = 1/4$$
$$\beta_{\mathsf{Pup}}(\hat{\mathcal{V}}) = 1/4$$

This identifies Mv as the most important action (unlike Shapley values) and correctly attributes zero value to Scn. The other two actions again are considered to be equally important.

Comparison Between the Two Options Now, given the two quite different results emerging from the two value calculations, one could naturally ask the question: which is a better way to attribute failure contributions? The two forms of calculation correspond to two ways of looking at the plan prefix that led to the failure. Shapley value requires us to look at all the actions and consider only ways in which we can permute them in a valid way. These look at counterfactual plan prefixes where all the plans were used. Banzhaf, on the other hand, looks at subsets of the plans that are part of the plan prefix. Under these counterfactuals, we are considering cases where certain actions are removed. The result is that, in our example, the Shapley values seem to identify the more recent action effects that finally push battery charge below the limit needed for the condition. In contrast, the Banzhaf values identify actions according to how much they contribute to the charge deficit. It would be useful to run user studies to establish which of these counterfactual cases are more intuitive to a user.

In addition to the intuition, we can also see a difference in computational overhead. If one were to use naive techniques, Shapley value calculation in the worst case could require iterating over $(k-1)!$ sequences in the worst case. However, in the case of Banzhaf, we only need to consider 2^{k-1} subsets. However, checking for the existence of a valid action sequence that uses all $k-1$ actions and results in the failure of $f_k \diamond_k M$ could be more expensive. We need to do additional work to evaluate which of the methods allows for more effective algorithms.

Conclusion

This paper presents a novel game-theoretic analysis for how one could explain plan failures, particularly in its more generalized form, in the context of numeric planning problems. Instead of presenting a single method, we wanted to present how this formulation could give rise to different ways one could provide explanations. As shown, through our example, we see how the two different attribution methods, namely Shapley values and Banzhaf values, could end up identifying extremely different candidates for the main cause of the failure. It is worth noting that other conditions could produce very different results, including cases where the choices converge. As such, if we are to identify which value may be a better fit for plan explanations, we might need to test whether specific assumptions made by the two methods line up with the user intuitions. For example, do people require a notion of efficiency in the context of blame attribution in the context of plan failures? Or, when considering alternatives, would they only consider permutations of the entire action sequences? While we briefly touched on some differences between explanations for planning and machine learning settings, there are more differences we didn't discuss. One of them is the notion of causal links. While in our example, we consider a case where many actions delete the preconditions of one action, there might also be cases where one action contributes causal links to many subsequent actions. Such relationships aren't usually found within machine learning contexts. Additionally, this scenario might also have asymmetric effects on the two values. Because one could foresee cases where such settings would reduce the number of possible winning coalitions more than the number of permutations. Finally, there is the question of computational complexity. As we discussed, there is a difference in the computations needed to calculate the two values. If people are open to both explanation types, we might want to prioritize based on the availability of more efficient algorithms.

Acknowledgements

The authors would like to thank Dr. Toby Walsh and Dr. Edith Elkind for helpful discussions and comments on the formulations and ideas used in the paper.

References

Chalkiadakis, G.; Elkind, E.; and Wooldridge, M. 2011. *Computational aspects of cooperative game theory*. Morgan & Claypool Publishers.

Driessen, T. S. 2013. *Cooperative games, solutions and applications*, volume 3. Springer Science & Business Media.

Lehrer, E. 1988. An axiomatization of the Banzhaf value. *International Journal of Game Theory*, 17(2): 89–99.

Maschler, M.; Zamir, S.; and Solan, E. 2020. *Game theory*. Cambridge University Press.

Sundararajan, M.; and Najmi, A. 2020. The many Shapley values for model explanation. In *International conference on machine learning*, 9269–9278. PMLR.

Von Neumann, J.; and Morgenstern, O. 1947. *Theory of games and economic behavior, 2nd rev*. Princeton university press.

Winter, E. 2002. The shapley value. *Handbook of game theory with economic applications*, 3: 2025–2054.

Appendix

Theorem 1. *The modified Shapley value defined in Equation 3 still satisfies the efficiency property, i.e., the value of individual players adds up to the value of the grand coalition*

$$\sum_i \phi_i(\hat{\mathcal{V}}) = \hat{\mathcal{V}}(\hat{\mathcal{N}})$$

Proof Sketch. For a given player i, we could rewrite Equation 3, such that it reads

$$\phi_i(\hat{\mathcal{V}}) = \frac{1}{|\Pi^N|} * \sum_{\pi \in \Pi^N} (\hat{\mathcal{V}}(\text{Pre}(\pi, i) \cup \{i\}) - \hat{\mathcal{V}}(\text{Pre}(\pi, i)))$$

That is, we rearrange the sum, now that it's explicitly over all permutations, and the function $\text{Pre}(\pi, i)$ gives the set of players that appear before i in the sequence π. Now moving on to the sum of these individual players, we see

$$\sum_i \phi_i(\hat{\mathcal{V}}) = \frac{1}{|\Pi^N|} * \sum_i \sum_{\pi \in \Pi^N} (\hat{\mathcal{V}}(\text{Pre}(\pi, i) \cup \{i\}) - \hat{\mathcal{V}}(\text{Pre}(\pi, i)))$$

Rearranging the sum we get

$$\sum_i \phi_i(\hat{\mathcal{V}}) = \frac{1}{|\Pi^N|} * \sum_{\pi \in \Pi^N} \sum_i (\hat{\mathcal{V}}(\text{Pre}(\pi, i) \cup \{i\}) - \hat{\mathcal{V}}(\text{Pre}(\pi, i)))$$

If we are to expand the inner sum, we will see that all terms except $\hat{\mathcal{V}}(\hat{\mathcal{N}}) - \hat{\mathcal{V}}(\emptyset)$, will remain. Since $\hat{\mathcal{V}}(\emptyset)$ is zero, we get

$$\sum_i \phi_i(\hat{\mathcal{V}}) = \frac{1}{|\Pi^N|} * \sum_{\pi \in \Pi^N} \hat{\mathcal{N}} = \hat{\mathcal{N}}$$

Hence proving the statement. □