Blame Attribution for Multi-Agent Pathfinding Execution Failures

Extended Abstract

Avraham Natan
Ben-Gurion University of the Negev
Be'er Sheva, Israel
natanavr@post.bgu.ac.il

Roni Stern
Ben-Gurion University of the Negev
Be'er Sheva, Israel
roni.stern@gmail.com

Meir Kalech
Ben-Gurion University of the Negev
Be'er Sheva, Israel
kalech@bgu.ac.il

ABSTRACT

When executing large Multi-Agent Path Finding (MAPF) scenarios, faulty events can occur over time and contribute to the overall degraded system performance. This raises the problem of how to attribute blame over the set of faulty events. **The first contribution** of this paper is to define this problem and propose the well-known Shapley value for solving it. **The second contribution** is an efficient approach for approximating Shapley values that is inspired by diagnosis concepts.

KEYWORDS

Multi-Agent Systems; Multi-Agent Pathfinding; Diagnosis; Blame Attribution

ACM Reference Format:
Avraham Natan, Roni Stern, and Meir Kalech. 2023. Blame Attribution for Multi-Agent Pathfinding Execution Failures: Extended Abstract. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 3 pages.

1 MOTIVATION

Multi-Agent Path Finding (MAPF) is a problem of finding non-conflicting paths for a group of agents from a set of starting points to a set of goal points [20, 21]. Notable real-world MAPF instances occur in automated warehousing [22] and automated parking [26]. Finding high-quality solutions to MAPF problems is NP-Hard or worse [14, 15], yet modern MAPF algorithms can plan paths for hundreds of agents.

The execution of such Multi-Agent Plans (MAP) rarely goes smoothly and often deviates from the plan. Such deviations may occur as a result of internal reasons (jammed wheel), external reasons (obstacles), or due to imprecise assumptions about the world [4, 12, 13]. Such deviations may lead to unacceptable degradation in overall system throughput. As an example, consider an automatic warehouse where worker robots are tasked to move items from the factory output to a fleet of trucks. A delay in one of the robots may cause it to interfere with another robot which in turn will interfere later with other robots, and so on. Eventually, this can cause a significant and unacceptable delay in loading the truck.

An important question to ask when a multi-agent system fails is "what is the root cause of the failure?" Prior work proposed diagnosis algorithms for multi-agent systems [8, 9] were designed to answer this question and localize the responsible faulty events. In this work we ask the complementing question: "how much did each faulty event contribute to the system failure?" Answering this question is also known as *blame attribution*. To motivate answering the blame attribution question, consider an accident (a system failure) in a multi-agent system of autonomous vehicles. Diagnosis algorithms may infer which defective vehicles are to blame for the accident, but will not determine how much each vehicle contributed to it. This is important in order to fairly decide how to split the compensation costs between the defective vehicles owners. In an autonomous warehouse setting, blame attribution can also be used to focus the efforts of warehouse maintenance teams.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

2 CONTRIBUTION

The first contribution of our paper is to formally define the blame attribution problem in the context of MAPF execution failures. We call this the **Blame Attribution for Multi-Agent Pathfinding Execution Failures (BAMPEF)**. There may be multiple ways to attribute blame, but in this work we propose to use the well-known Shapley values [18, 19]. Shapley values are used in moral philosophy [11, 24], law [3], politics [2, 7], and other areas [5, 6, 10].

Informally, the goal of Shapley values calculation is to determine the division of power among a group of members. The approach for calculating this division is by using the *marginal contribution* of each member to the various subsets of the member group. A formal definition can be found in many forms in the literature [17, 23, 25].

Calculating Shapley values is computationally costly, as the calculation must consider all the possible subsets of a given set of faults, which is exponential. **The second contribution** of this paper is a fast method to approximate the Shapley value that we call **Diagnosis-Directed Blame Attribution (*DDBA*)**. *DDBA* uses concepts from the field of Model-Based Diagnosis [16] to identify which subsets of fault events are sufficient to consider to obtain an effective approximation for the Shapely Value. Limiting the Shapley calculation only to consider these subsets of faulty events significantly reduces the run-time. For instance, in an example of 13 fault events it took 1.62 seconds on average, while calculating the full Shapley values for that example took 36.2 seconds.

3 METHODOLOGY

In this work, we use Shapley values to distribute blame among the faults. To that end, we look at an execution of a MAP as a game where the set of members is the set of accelerations or delays that occurred in the plan execution, which we denote as Fault Events (E). Such execution will result in a degree of degradation in the system's performance. We assume a value function v, that given E, calculates the value of system degradation. This allows us to use Shapley values to determine the division of blame of the system degradation among the Fault Events E. Formally, Definition 1 (**Shapley Values for BAMPEF**). *Given a set E of n Fault Events and a value function $v : 2^E \rightarrow \mathbb{R}$, the Shapley Value for Fault Event e is:* $\phi_e(v) = \sum_{E' \subseteq E \setminus \{e\}} \frac{|E'|!(n-|E'|-1)!}{n!} \left[v(E \setminus (E' \cup \{e\})) - v(E \setminus E') \right]$

Using Shapley values requires to process the entire power set of E. This may lead to exponential computational time. To address this, we improve Shapley values calculation using diagnosis concepts. Diagnosis processes aim to identify the root cause of a failure in a system. In our domain, the root cause is the Fault Event(s) that caused the degradation in the system's performance. To this end, we define first the concept of Useful Repair. A Useful Repair is a subset of the Fault Events set, that improves the system performance when the system is simulated without having those faults. Formally:

Definition 2 (**Useful Repair**). *Given a set E of n Fault Events and a value function $v : 2^E \rightarrow \mathbb{R}$, the set of Useful Repairs is $\Omega = \{E' \subseteq E : v(E \setminus E') < v(E)\}$.*

Once the set Ω is calculated, we calculate the Shapley values of the Fault Events with respect to each $\omega \in \Omega$. In that way, for every Useful Repair we calculate how much every participating Fault Event contributed to the system repair. We call this approach **Diagnosis Directed Blame Attribution (*DDBA*)**. We extend the definition of Shapley value to consider the set ω:

Definition 3 (**Shapley Value for BAMPEF w.r.t ω**). *Given a set $\omega \subseteq E$ of n Fault Events and a value function $v : 2^E \rightarrow \mathbb{R}$, the Shapley value w.r.t ω for Fault Event e is defined as follows:*

$$\phi_e^{\omega}(v) = \sum_{\omega' \subseteq \omega \setminus \{e\}} \frac{|\omega'|!(n - |\omega'| - 1)!}{n!} \left[v(E \setminus (\omega' \cup \{e\})) - v(E \setminus \omega') \right]$$

Once the Shapley values of the Fault Events have been calculated with respect to each ω, we aggregate them to receive the final values.

At this point Ω may still be big - there may be a lot of Useful Repair sets. Shapley would run on all $\omega \in \Omega$, and this might lead to long run-times. In order to reduce this run-time, we propose to decrease the size of Ω by considering only $\omega \in \Omega$ with cardinality up to a number k. We denote the resulting smaller set as Ω'. To that end, we iterate over the different cardinalities $i \in [1, ..., k]$, and for each cardinality we compute Shapley values of $\omega \in \Omega'$ with cardinality i. Finally, we aggregate those values over all the Fault Events in Ω' to achieve an approximation to the Shapley value.

4 EVALUATION

For experiments, we generated instances with ranging number of plan lengths $y \in \{8, 10, 12\}$, agents $x \in \{8, 10, 12\}$, faulty agents $f \in \{3, 4, 5\}$, fault probabilities $p \in \{0.5, 0.7, 0.9\}$. We compared our algorithm (denoted ***DDBA***) with the traditional Shapley values calculation (denoted ***Gold***), and an existing random sampling algorithm [1], where the subsets of the Fault Events for the shapley calculation are selected randomly (denoted ***Random***). As a comparison metric, we use Euclidean Distance to measure the distance between the Shapley values of the approximate approaches and the Shapley gold standard (denoted *error*). In addition, we measure the run-time (in seconds) of the methods.

Useful Repair Cardinality		1	2	3	4	5	6	7	8
Average	Random	0.146							
Error	DDBA	0.355	0.229	0.138	0.107	**0.089**	**0.076**	0.069	0.066

Table 1: Error of *Random* and *DDBA*, with the increase of the useful repair cardinality (k).

Useful Repair Cardinality		1	2	3	4	5
Average	Gold	7.024				
Runtime	Random	1.074				
(Seconds)	DDBA	**0.006**	**0.036**	**0.167**	**0.588**	2.077
Average	Random	0.151				
Error	DDBA	0.376	0.257	0.170	**0.130**	**0.107**

Table 2: Average runtime and error of *Random* and *DDBA* with the increase in the useful repair cardinality.

Fault Events		6	7	8	9	10	11	12	13
Average	Random	0.147	0.146	0.148	0.149	0.153	0.154	0.153	0.156
Error	DDBA	**0.112**	**0.113**	**0.120**	**0.127**	**0.133**	**0.140**	**0.146**	**0.146**
Average	Gold	0.052	0.152	0.334	0.737	1.745	4.449	12.505	36.215
Runtime	Random	0.062	0.129	0.256	0.467	0.718	1.239	2.174	3.546
(Seconds)	DDBA	**0.038**	**0.090**	**0.180**	**0.363**	**0.517**	**0.785**	**1.107**	**1.623**
DDBA Useful Repairs		34.84	58.75	92.37	138.64	198.32	272.80	367.96	476.65

Table 3: Results for different number of faulty events.

Table 1 shows the error of *DDBA* for Useful Repair cardinality of $k = 1, \ldots, 8$, for BAMPEF instances with 12 agents, plans of length 12, 5 faulty agents, 0.9 fault probability and 10 Fault Events. The error decreases fairly fast until $k = 5$, and then decreases significantly slower. Since increasing k means higher runtime, we limited k to be at most 5 in the remaining experiments.

Table 2 shows the runtime and error for *DDBA*, *Random*, and *Gold*. As expected, *DDBA* is much faster than *Gold*. We also observe that *DDBA* is faster than *Random* for $k \leq 4$. The error of *DDBA* is higher than *Random* for $k < 4$, and lower when $k \geq 4$. Thus, our results confirm the expected trade-off provided by the Useful Repair cardinality k between runtime and error. In our experiments, however, setting $k = 4$ provides an effective middle-ground between runtime and error. Hence, in the next results, we fixed k to 4.

Table 3 presents the results of *DDBA*, *Random*, and *Gold* for varying number of faulty events (*fe*). First, the runtime of *Gold* increases exponentially with *fe*. The runtime of *Random* also increases, but at a much lower rate than *Gold*, while the runtime of *DDBA* is even lower. For instance, while considering *fe* = 13, *DDBA* runs 2 times faster than *Random* on average, and 50 times faster than *Gold*. This shows the efficiency of *DDBA* when considering high *fe*. Second, the error of both *Random* and *DDBA* increases very slightly with *fe*, which suggests that *Random* and *DDBA* are not influenced much by *fe*. This means that *DDBA* is scalable for larger amounts of fault events. In addition, the error of *DDBA* is lower than the error of *Random*. Specifically, this difference is higher when considering low *fe*. This suggests that for small systems, *DDBA* is preferable over *Random*. Third, the runtime of *DDBA* is strongly affected by the number of useful repairs it considers. To highlight this, Table 3 also shows the number of useful repairs considered by *DDBA* for a different *fe*. Indeed, the runtime of *DDBA* is correlated with the number of useful repairs, which increases with *fe*. For example, for *fe* = 8, with cardinality up to $k = 4$, the empirical number of useful repairs is $\binom{8}{4} \cdot 2^4 = 1120$, while the actual number is 92.37 on average. This gap is because many subsets of the fault events are not useful repairs.

ACKNOWLEDGMENTS

This research was funded by ISF grant No. 1716/17, and by the ministry of science grant No. 3-6078.

REFERENCES

[1] Javier Castro, Daniel Gómez, and Juan Tejada. 2009. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research* 36, 5 (2009), 1726–1730.

[2] GN Engelbrecht and AP Vos. 2009. On the use of the shapley value in political conflict resolution. *Scientia Militaria: South African Journal of Military Studies* 37, 1 (2009).

[3] Samuel Ferey and Pierre Dehez. 2016. Multiple causation, apportionment, and the Shapley value. *The Journal of Legal Studies* 45, 1 (2016), 143–171.

[4] Gordon Fraser, Gerald Steinbauer, and Franz Wotawa. 2005. Plan execution in dynamic environments. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 208–217.

[5] Christopher Frye, Colin Rowat, and Ilya Feige. 2020. Asymmetric Shapley values: incorporating causal knowledge into model-agnostic explainability. *Advances in Neural Information Processing Systems* 33 (2020), 1229–1239.

[6] Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*. PMLR, 2242–2251.

[7] Franz Hubert and Svetlana Ikonnikova. 2003. Strategic investment and bargaining power in supply chains: A Shapley value analysis of the Eurasian gas market. *Humboldt University Berlin* (2003).

[8] Meir Kalech and Avraham Natan. 2022. Model-Based Diagnosis of Multi-Agent Systems: A Survey. In *AAAI Conference on Artificial Intelligence*. 12334–12341.

[9] Eliahu Khalastchi and Meir Kalech. 2019. Fault detection and diagnosis in multi-robot systems: a survey. *Sensors* 19, 18 (2019), 4019.

[10] Richard TB Ma, Dah Ming Chiu, John CS Lui, Vishal Misra, and Dan Rubenstein. 2007. Internet Economics: The use of Shapley value for ISP settlement. In *Proceedings of the 2007 ACM CoNEXT conference*. 1–12.

[11] Moshe Mash, Roy Fairstein, Yoram Bachrach, Kobi Gal, and Yair Zick. 2020. Human-computer coalition formation in weighted voting games. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 6 (2020), 1–20.

[12] Jonathan Morag, Ariel Felner, Roni Stern, Dor Atzmon, and Eli Boyarski. 2022. Online Multi-Agent Path Finding: New Results. In *Proceedings of the International Symposium on Combinatorial Search*, Vol. 15. 229–233.

[13] Avraham Natan and Meir Kalech. 2022. Privacy-aware Distributed Diagnosis of Multi-Agent Plans. *Expert Systems with Applications* 192 (2022), 116313.

[14] Bernhard Nebel. 2020. On the computational complexity of multi-agent pathfinding on directed graphs. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 30. 212–216.

[15] Bernhard Nebel, Thomas Bolander, Thorsten Engesser, and Robert Mattmüller. 2019. Implicitly coordinated multi-agent path finding under destination uncertainty: Success guarantees and computational complexity. *Journal of Artificial Intelligence Research* 64 (2019), 497–527.

[16] Raymond Reiter. 1987. A theory of diagnosis from first principles. *Artificial intelligence* 32, 1 (1987), 57–95.

[17] Alvin E Roth. 1988. Introduction to the Shapley value. *The Shapley value* (1988), 1–27.

[18] L Shapley. 1953. Quota solutions op n-person games1. *Edited by Emil Artin and Marston Morse* (1953), 343.

[19] Lloyd S Shapley and Martin Shubik. 1954. A method for evaluating the distribution of power in a committee system. *American political science review* 48, 3 (1954), 787–792.

[20] Roni Stern, Nathan R Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Satish Kumar, et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Twelfth Annual Symposium on Combinatorial Search*.

[21] Nathan R Sturtevant. 2012. Benchmarks for grid-based pathfinding. *IEEE T-CIAIG* 4, 2 (2012), 144–148.

[22] Hongtao Tang, Xiaoya Cheng, Weiguang Jiang, and Shouwu Chen. 2021. Research on Equipment Configuration Optimization of AGV Unmanned Warehouse. *IEEE Access* 9 (2021), 47946–47959.

[23] Stelios Triantafyllou, Adish Singla, and Goran Radanovic. 2021. On Blame Attribution for Accountable Multi-Agent Sequential Decision Making. *Advances in Neural Information Processing Systems* 34 (2021), 15774–15786.

[24] Menahem E Yaari. 1981. Rawls, Edgeworth, Shapley, Nash: Theories of distributive justice re-examined. *Journal of Economic Theory* 24, 1 (1981), 1–39.

[25] H Peyton Young. 1985. Monotonic solutions of cooperative games. *International Journal of Game Theory* 14, 2 (1985), 65–72.

[26] Jiawei Zhang, Zhiheng Li, Yidong Li, and Hairong Dong. 2021. A bi-level cooperative operation approach for AGV based automated valet parking. *TR_C* 128 (2021), 103140.