Advanced Analytics for Space-enabled Kill-webs (AASK)

Dr. David Hillstrom
Infinity Labs, LLC.
Dayton, OH
david.hillstrom@i-labs.tech

ABSTRACT

This paper presents a method for applying Markovian 'fault diagnosis' from model-based engineering to reveal nuance in cause and effect within complex and probabilistic mission-level simulations. The approach is designed to address challenges introduced in analysis of Mosaic Warfare scenarios. Mosaic Warfare refers to the assembly of many warfighting platforms into a larger force package that leverages complexity and turns it into an asymmetric advantage. Mosaic Warfare presents a challenge in analysis with mission-level virtual experiments as the layering of forces and technologies results in a complex system of systems. As complexity grows, the importance of individual contributions becomes harder to identify at the macro scale. Infinity Labs presents a method for distilling this complexity with an approach that goes beyond single platform analysis by mathematically linking webs of platforms together with vectors of relevant data. Each vector describes a state, or situation, in the mission. With this, new analysis methods become viable as the connection and transitions between these states reveal specific moments within a scenario where vulnerabilities in the kill-web occur. In the scope of a mission, a vulnerability is a negative event such as: loss of an asset, a failed sensor detection, an un-received message, or any event which diverted the scenario towards failure. A brute force application of Markovian analysis on mission-level studies would be impossible on today's computers, but here, it is enabled by the development of a novel and scalable technique for extracting, organizing, and clustering mission-level data. This paper presents an overview of initial results where a simple virtual mission is evaluated using these methods. Going forward, this method helps focus technology and strategy development efforts where they are needed by directly highlighting vulnerabilities in the mission.

ABOUT THE AUTHORS

Dr. David Hillstrom is a Principal Investigator at Infinity Labs, LLC. He is a scientist with 11 years of research experience in academia at The Ohio State University and subsequently 2 years with industry at Infinity Labs. In that time, he conducted complex fundamental research involving system modeling, computational fluid dynamics, spray performance, systems engineering, and combustion. In 2019, he helped found OSU-SIMCenter's Advanced Framework for Simulation, Integration, and Modeling (AFSIM) mission-level laboratory in cooperation with Professor Shawn Midlam-Mohler and the Air Force Research Laboratory (AFRL). David went on to lead the research efforts of the laboratory from 2021-2023 with a focus on researching the application of novel model-based engineering techniques for analyzing complex missions. As principal investigator, David demonstrated Markovian analysis on complex many-on-many missions (~200 platforms) in an industry first at that scale. Beyond this, he led research efforts to quantify uncertainty within AFSIM in a step towards verification and validation (V&V). Later, he worked with AFRL to leverage this uncertainty data to develop new mission performance optimization techniques which consider the statistics of data.

Advanced Analytics for Space-enabled Kill-webs (AASK)

Dr. David Hillstrom
Infinity Labs, LLC.
Dayton, OH
david.hillstrom@i-labs.tech

INTRODUCTION

The concept of 'Mosaic Warfare' refers to the assembly of many individual warfighting platforms into a larger force package that leverages complexity and turns that into an asymmetric advantage (DARPA, 2018). Space plays a critical role in this strategy by supporting and augmenting terrestrial force designs, enabling them to connect and collaborate in new ways that evolve their capability. At the scale of mission-level simulations, these connections create complex webs of interactions that are difficult to parse into actionable performance information for a particular platform, technology, or moment. Current methods generally leverage highly averaged metrics of performance logged at the end of the mission (Bernal, 2020). This captures general performance trends but does not explain the cause-and-effect within the web of platform interactions that would help focus technology and strategy development efforts. For example, a new technology may generate 20% better performance in one area of the kill-web, but 20% worse performance in another. The final performance metric would be nearly unchanged, but exploring the results in finer detail presents a better understanding of the technology's impact on the mission and provides the means to target additional development towards mitigating the specific area where performance was degraded.

AFSIM as the Virtual Testing Ground

In the wargaming simulation space, the AFSIM software serves as a virtual testing ground to simulate missions with a large variety of interacting and collaborating agents (West & Birkmire, 2019). These agents are represented by a 'digital twin' of their physical self with varying levels of fidelity (e.g., airborne early warning and control (AWACS), surface-to-air missile (SAM) sites, drones, and satellites). Within the software, analysts simulate missions by collecting these agents together and controlling them to represent a larger scenario of interest, which could include a side-vs-side engagement. These systems are often implemented to simulate a kill-web, and interruptions to the kill-web, from start to finish in pursuit of removing some threat. A space-layer can be added to directly test and compare its impact on mission performance. Although, while the software can execute extremely complex missions, the DoD, industry, and academia are actively researching methods for more comprehensive analysis capability on those missions. Often, the scenario results are reduced to a simple bar-chart with error-bars as the wealth of available data is difficult to package and present such that an analyst can leverage it for understanding or to make decisions. Despite this, AFSIM serves as a common testbed for mission-level studies and technology tests as the software is government owned, open-source, and nonproprietary, making the sharing of results and conclusions easier within the defense community.

Probabilistic Outcomes in Mission-Level Simulations

Within mission-level modeling frameworks like AFSIM, the scale of simulations often requires models which have reduced complexity to enable results at the speed of relevance. In an example model simplification, many physically complex interactions, such as a weapon hitting a target, are resolved through probability (Figure 1). The details of that probability are informed by external experiments or high-fidelity simulation (West & Birkmire, 2019). For example, repeated physical testing of weapon A on platform X to determine a probability of kill for weapon A against platform X against a variety of independent design variables. After those tests are complete, the results are digested into a probability of kill tables leveraged in the mission-level simulation. Alternatively, the probability may come from program requirements for a concept technology that does not have a physical or digital prototype. In this case, the probability will be set by the required capabilities of the technology in consideration. This allows testing the impact of a theoretical technology on the outcome of a mission before detailed engineering development.

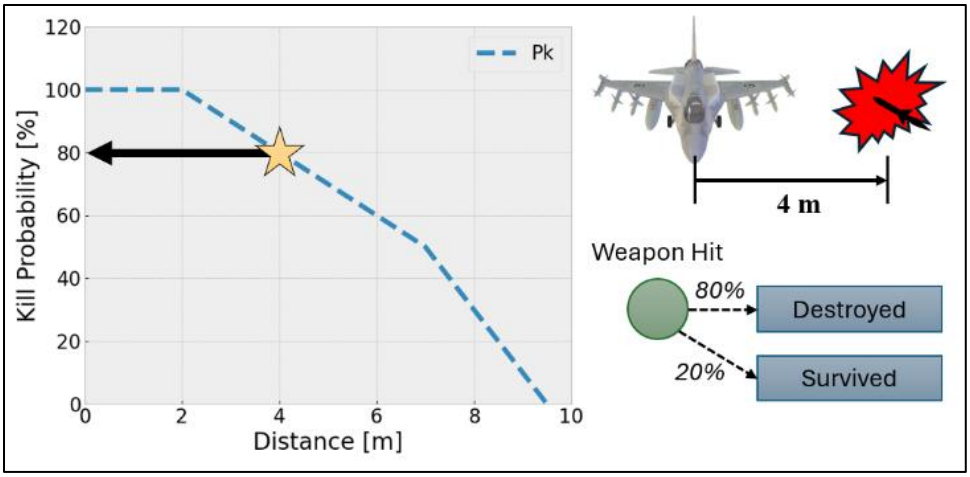

Figure 1. Sample Mission-Level Physical Interaction Modeled Through Probability Creating a Branching Future Based on the Outcome.

This probability driven format, necessitated by mission-level complexity, provides an avenue to explore the implication of technology changes in new ways. Each probability draw provides a fork in the simulation's path. A better technology, or strategy, may influence that fork towards a better path at the more critical moments. Leveraging this in analysis will yield a more nuanced perspective on what the technology affected, especially as the complexity of the mission grows and the importance of individual contributions become harder to identify at the macro scale. A deviation in the simulation path may also result from a mission participant making a different decision or doing a different action, not just physics approximating probability draws. The mathematical backbone to represent this variety in outcomes is present within the AFSIM software but is often viewed as a hinderance to be overcome or, in the worst case, ignored. By leveraging concepts in model-based engineering (MBE), machine-learning, and statistics, there is an opportunity to better embrace AFSIM's branching nature to develop more insightful analysis. This analysis is sourced from techniques which shed light on the specifics of where, when, how, why, and with what probability specific actions lead to specific results.

QUANTIFYING CAUSE AND EFFECT

The analysis goal is to quantify, visualize, and understand the interactions and outcomes from all platforms, from any given moment. Through this, all sequences of action → consequence, or cause → effect, are captured. One method to enable this is to develop a Markov Chain of the mission (Billingsley, 1961). A Markov Chain is a stochastic process that provides the probability of what happens next based on the state of affairs now. Thus, if the mission state can be described, a state transition matrix of probabilities provides a means to define the available branching futures. The determination of state transition probabilities can be acquired through robust sampling of the simulation by executing the mission with a sufficient number of random seeds and with all relevant decisions, strategies, and actions.

A Markov Chain provides the means to infer information on transition probabilities for one long observation of states $\{x_1, x_2, \dots, x_n\}$. Here, the states x_n represent the state of a mission, and its included kill-webs, at time n. The evolution of the state system is governed by a stationary stochastic $s \times s$ matrix of probabilities p_{ij} referred to as the state transition matrix. The Markov Chain then defines the probability of reaching any state x_{m+n} from any current state x_m as:

$$P\{x_{m+n} = j \;||\; x_m = i\} = p_{ij}^{(n)} \qquad (1)$$

Three sample mission simulations and the state observations are shown in Table 1. In the first simulation, seed 1, all observed states are new so they are simply ordered sequentially. In the second simulation, seed 2, similar states to seed 1 are observed again, but in a new order, until ultimately new state observations are made at the 6th and 7th moment given the state labels of 7 and 8. Finally, the third simulation does not observe a new situation but simply a different ordering of states which were observed in the first two seeds.

Table 1. Observed State Progression for Three Different Random Seeds.

Run #	State 1	State 2	State 3	State 4	State 5	State 6	State 7
Seed 1	1	2	3	4	5	6 [end]	
Seed 2	1	3	2	5	6	7	8 [end]
Seed 3	1	2	4	6	5 [end]		

Leveraging the observed state progressions, the probability of transitioning from one state to any other state can be calculated. Consider the transition from state 1. Two out of three observations move to state 2, but one observation moves to state 3. Using this, $P_{1,2} = 66\%$ and $P_{1,3} = 33\%$. State 1 is not observed to move to any state but 2 or 3 so their probabilities will be 0% (i.e., $P_{1,4} = 0\%$). Using this, the full state transition matrix is viewable in Table 2.

Table 2. State Transition Matrix from the Three Observed Seeds.

		To							
		1	2	3	4	5	6	7	8
From	1		0.67	0.33					
	2			0.33	0.33	0.33			
	3		0.50		0.50				
	4					0.50	0.50		
	5					0.33	0.67		
	6					0.33	0.33	0.33	
	7								1.00
	8								1.00

The Kill Web State

To enable the Markovian study, it is necessary to define a mission state. A platform state would typically include a variety of data features relevant to a particular study like speed, heading, position, *etc*. The platform state is a vector of data where each dimension reflects an important parameter for that study. However, to enable a Markovian evaluation of a kill-web, the mission state must consider the collection of all involved platforms. A straightforward approach to achieving this is to describe all data features of interest for an individual platform and then append all platforms together into a larger single vector. If these values are determined for all platforms, and all platforms are appended together, that vector then represents the mission 'state' at a given moment. Any time a feature value is changed the mission shifts to a new state.

A sample mission state progression would be the traditional kill-chain Find, Fix, Track, Target, Engage, Assess (F2T2EA). As the kill-chain participants change their activity, or acquire new orders or information, their location in the chain shifts. The F2T2EA kill-chain represents a 1-to-1 transition of states but is highly simplified (Figure 2). For most real mission states, there will be many possible outcomes for any given state driven by participant decisions, strategies, events, or random draws from probabilistic physical models.

Figure 2. Sample State Sequence for the F2T2EA Kill Chain.

Comparing Kill Webs with Mission-level Markov Chains

With sufficient sampling of the mission, all reasonably probable state sequences are identified. Packaging the state transition matrix as a Markov Chain presents the information as a web of state connections describing all possible futures for any given moment. If two technologies or strategies are simulated, then two different Markov Chains are created and can be compared. As long as there is uniformity in the state definitions, this presents an opportunity to explore in detail where new technology had an impact. Consider Figure 3 where two kill-web systems are compared. For simplicity, the Markov Chains have the starting state and a successful and unsuccessful terminal state clearly identified. In this comparison, the improved system changed the outcome probabilities for state 2, which directly impacted the trajectory of the mission and ensured a higher chance of success. In this simplified example, State 2 represents a key vulnerability where the wrong outcome will result in kill-web failure. This comparison is the target output of the Phase II program in 2027. Here, Infinity presents several challenges associated with representing mission-level simulations with state vectors and presents methods to overcome them in progress towards this ultimate comparison of the Markovian directed graphs.

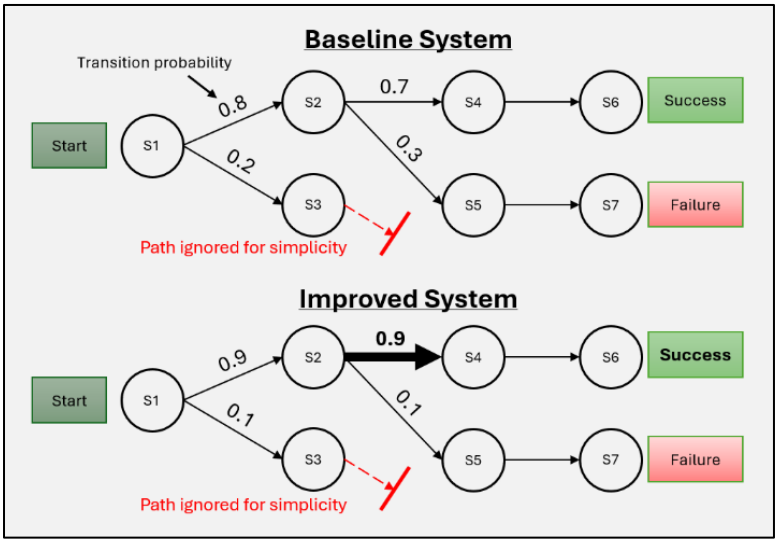

Figure 3. Comparing Two Technologies Through Markov Chains. This target Phase II output highlights specific moments where the technology improved the chance of success.

SCALING TO MANY PLATFORMS

Applying Markovian style analysis has been performed successfully by the Air Force Institute of Technology with students exploring the application of a Markov Decision Process in AFSIM engagements (Palm, 2023). However, these reinforcement-learning analyses are limited to 1v1 or 2v2 scenarios due to processing times and memory requirements. To visualize the scalability concerns of an expanding mission state vector, consider a state vector which only includes binary features (0 or 1). In this case, the theoretical max number of states S_{MAX} is then:

$$S_{MAX} = 2^{a*b} \qquad (2)$$

Where a is the number of platforms and b is the number of features. This is the best-case scenario for raw data memory consumption as it represents 1 bit per feature. Still, in this form, the required memory can be extreme for all possible states (Figure 4). With only six features, the potential storage capacity for only twelve platforms matches the expected global data usage in 2025 (Reinsel, D., Gantz, & Rydning, 2018). In practice, the realized state space is significantly smaller as not all possible states are observed. Even so, there is an opportunity to design the state vector to minimize memory scaling.

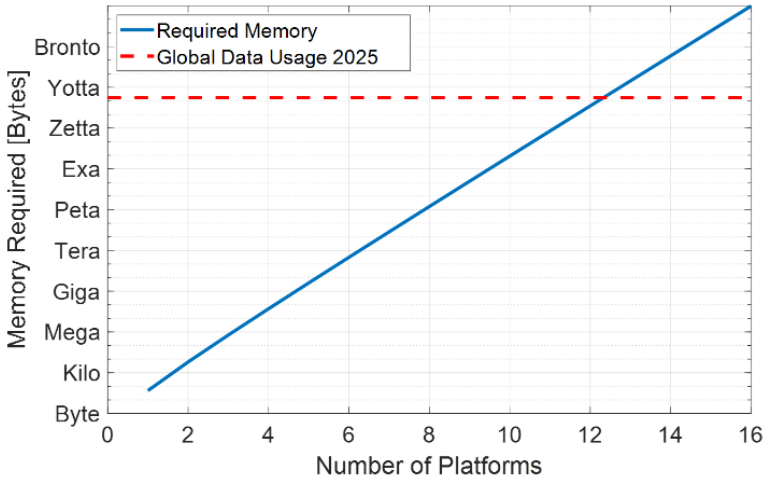

Figure 4. Theoretical Max Memory Requirement for 6 Features.

Feature Simplicity

The first key component to reducing memory requirements of the state vector is in feature simplicity. In general, all features must be binary, with limited use of integers as necessary. This requires an alternative approach to classifying the state of a platform as it is no longer appropriate to include typical continuous values as absolute position, velocity, throttle, thrust, fuel, *etc.* Instead, the features leveraged represent status inquiries of the system that reflect the goals of the mission. For example, consider an unmanned bomber approaching an integrated air defense system (IADS). For the bomber, instead of logging position and velocity, ask a targeted inquiry: 'are you in range?' The feature is then 'in range' with a binary value of true or false. In this way, the state vector is made up of simplified quantities which represent objectives for the participants. An example of continuous, integer, and binary features are demonstrated in Table 3. When digested into a sequence of states, the flips in data tell the story of the mission such as 'fly north' → 'ingress' → 'target SAM-3' → 'target SAM-7' → egress. With many simulation samples of the mission, diversions in the story and their likelihood of finding mission success or failure are represented mathematically. In the prior example, this could manifest as SAM-3 being a target priority that contributed towards success. If SAM-7 was targeted first, the probability of mission failure, as defined in the future available states within the Markov Chain, go up. If using binary only features, it has been demonstrated that a full mission evaluation and design of experiments including nearly 500k simulations can be performed with two hundred platforms while still fitting entirely in memory on a high performance laptop (Hillstrom, Midlam-Mohler, Jankord, & Isaman, 2024).

Table 3. Example Features for an Arbitrary Platform. Continuous features result in an infinite number of states whereas integer and binary features are finite.

Feature	Type	Lower Limit	Upper Limit	State Count
Speed	Continuous	0	1130 mph	∞
Weapon Count	Integer	0	5	6
Armed?	Binary	0	1	2

Feature Compression

In some cases, it may be necessary to maintain higher fidelity on a data feature beyond true and false. For example, platform speed, where the analyst may desire to frame their interrogation of mission and kill-web performance to include speed differences between platforms if a key technology being tested directly affects this. For example, in counter to an incoming weapon, a platform may increase speed to improve its chance for survivability. If speed is not logged, the survivability of the platform may seem random, when in fact, increasing speed was the direct action that prevented a kill. The challenge is that speed is continuous. For the purposes of the state vector, is 100 mph different enough from 100.1 mph to constitute a new kill-web state? Other than the memory scaling requirements, this

granularity in state tracking makes it difficult to compare kill-webs as two mission states, one with a platform at 100 mph, and the other at 100.1 mph, would have different labels. The limiting case of too much granularity results in every possible simulation translating into an entirely unique sequence of states. If 1000 simulations are sampled at this limit, the Markov Chain is 1000 parallel lines, preventing comparison. To ease the capability of comparing complex Markov Chains, discrete differences in feature values must be limited to only mission-relevant quantities. In this case, 100.1 vs 100.0 mph is probably not valid, but perhaps 100 vs 200 mph is.

To support comparison of continuous quantities, the Markovian analysis approach enables feature compression. Through this, continuous variables are reduced to discrete bins labeled with integers. To continue with the speed example, a user specified step is used to discretize the space. If the user specifies a step of 100 mph, then the bins would be 0-100, 100-200, 200-300, *etc.* resulting in integer bins of 1, 2, 3, *etc.* Leveraging this, the analysis will differentiate kill-web states based on platform speed, but only at the fidelity of the bin size. The value within the state vector is then limited to a bin integer within the observed speed envelope of the platform. While this abstracts the detail of the speed value, it still provides the means to distinguish one platform as faster than another. It is necessary to take care in designing the size of the bins to ensure the differences in technology manifest in new states. If one technology operated at 101 mph and the other at 199 mph, these settings would yield a speed value of 2 for both platforms giving them the same label within the Markov Chain and not differentiating them as different situations.

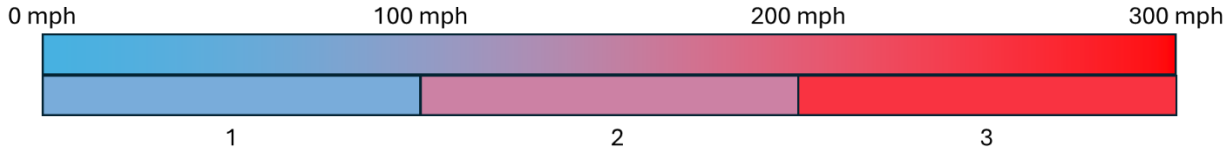

Figure 5. Platform Speed Compression Example. The continuous 0-300 mph space is reduced to three bins in steps of 100 mph reducing the number of possible states in this kill-web.

Merging and Grouping Platforms

To extend the Markovian analysis to work with kill-webs containing hundreds of participants, feature simplicity and feature compression may not be enough on their own. The final contribution to state vector design which enables scaling to Mosaic Warfare relevant missions that include space assets is the merging and grouping of platforms. The objective is to mimic situational-awareness style grouping which may consider some platform groups as one entity. For example, a swarm of drones, or a flight of bombers, may be thought of as one entity for the purpose of decision making in a given environment. In terms of space assets, the GPS constellation may be considered one entity that is either sufficiently providing location, or not. Three example criteria used for grouping are position, function, or communication. An example of grouping performed using these criteria is shown in Figure 6.

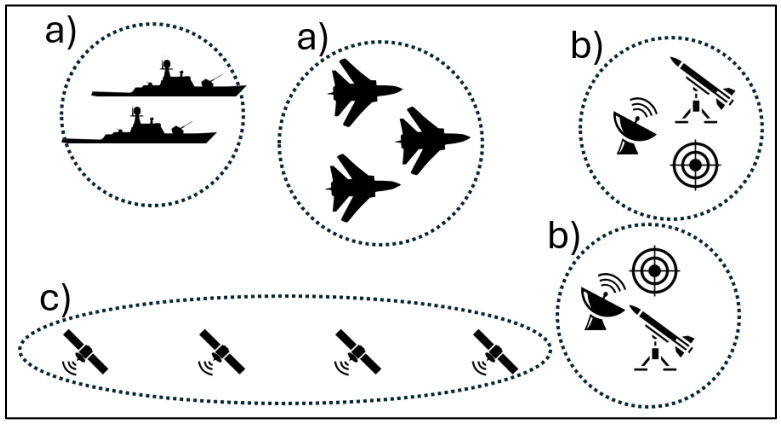

Figure 6. Platform Merge and Grouping Strategies. a) position and function. b) position and communication. c) communication and function.

It is not necessary to use a combination of criteria. Platforms could be grouped or merged based on position, communication, or function on their own. The criteria provide data which is leveraged in an algorithm of choice by the analyst to determine the levels of grouping. For position, one example would be k-means clustering based on location in the scenario (Hartigan & Wong, 1979). Throughout the mission, the platforms that were, on average, near each other can be flagged for grouping. In this way, the platforms would be regarded as one entity in the state vector. Communication and function groups may be specified directly by the analyst or observed naturally by linking webs of communicating platforms together when executing tasks within the kill-web. Once the platforms to merge are identified, it is necessary to decide how their data features are combined.

	Platform 1			Platform 2			Platform 3		
Feature Num.	Feature 1	Feature 2	Feature 3	Feature 1	Feature 2	Feature 3	Feature 1	Feature 2	Feature 3
Data	Weapons	Detected	Fuel	Weapons	Detected	Fuel	Weapons	Detected	Fuel
State 1 (S1)	3	False	1	5	False	2	4	True	2

Merge Platforms 1-3 into one entity

	Platform 1, 2, and 3		
Data	Weapons	Detected	Fuel
Strategy	Sum	Any	Min
State 1 (S1)	12	True	1

Figure 7. Grouping Example on Three Platforms. The merge strategy can be specified per feature depending on the goals of the analysis and intended observations of the kill-web for comparison.

The merge strategy is analyst defined and depends on the data feature and the analysis goals in question. Using a statistical quantity is common with the examples mean, median, mode, min, or max. Alternatively, an algorithm can be used for their combination, such as their sum or difference. For example, consider the weapon count of a group of bombers tasked with destroying a single target. If all bombers are merged into one entity, it may be appropriate to sum the total weapon count between them. An example of merging three platforms together, using different strategies for each feature, is seen in Figure 7.

Consider the impact this has on the feature for number of weapons. For simplicity, consider this feature in isolation on the three platforms, then the max number of states is:

$$S_{MAX} = (R_{MAX} + 1)^a \qquad (3)$$

Where R_{MAX} is the maximum number of weapons carried by a platform and a is the number of platforms. In this case, there are up to five weapons and there are three platforms yielding a maximum number of states as 216. If we reduce the feature for all three platforms to sum together into one merged entity, the maximum number of states becomes:

$$S_{MAX,sum} = aR_{MAX} + 1 \qquad (4)$$

In Equation (4), the maximum number of states in this example is 16, significantly smaller than the 216 observed before grouping. Importantly, the scaling of the state space complexity is linear in (4) as compared to exponential in (3). The complexity reduction observed from the 'any' strategy on Feature 2 and the 'Min' strategy on Feature 3 are even more aggressive than the Sum demonstration above. With this final state vector design component, the state space of a complex mission can be sufficiently restricted for computation and analysis.

IDENTIFYING KILL WEB VULNERABILITIES

A key challenge for leveraging Markovian style analyses like this on a mission is automatically identifying states of importance, like S2 within Figure 3. For a mission of relevant complexity which approaches Mosaic Warfare, the Markov Chain is untenably large for manual navigation and comparison. Therefore, it is necessary to develop an algorithm to separate event chains which lead to success and those which lead to failure and identify the most important

moments in time where the mission typically committed towards one direction or the other. The Markov Chain is a directed graph, providing the means to explore graph theory, connectivity, node manipulation, and other methods for understanding.

Test Mission Description

One method for identifying vulnerabilities is through scoring the underlying state vectors measured within the simulation. To demonstrate the application of the scoring method to identify vulnerable moments in a mission, a simplified IADS mission is presented. The scenario laydown includes the two sides blue and red. Blue is defensive with a collection of IADS units and command and control (C2) systems. In Blue's territory is a collection of high value targets (HVTs) and defensive counter-air platforms. Red is the offensive force with unmanned combat aerial vehicle (UCAV) bombers. To aid the offensive, Red has stand-off jammers to reduce Blue's effectiveness and their own required C2 units. The mission objective is for Blue to defend its priority targets from Red's offensive UCAV bombers.

Scoring States to Identify Vulnerabilities

The objective is to develop a method to score states such that each mission experiment, and the involved kill-webs, can be graded on performance over time. Through this, periods of strong negative gradients can provide insight into poor regions of performance where a vulnerability may exist. The vulnerability can be a poor decision, failing technology, or a non-optimal strategy. A simplified example of this is seen in Figure 8b. In this qualitative plot, the red curve has a step down in score near eight minutes. This failure is likely preceded by a vulnerability in the kill-web that can give insight into how to improve performance.

a)

Scoring Configuration			
Feature	**Score Weight**	**Blue Impact**	**Red Impact**
Platforms Tasked	1	Good (+)	Bad (-)
Tracks Identified	0.4	Good (+)	Bad (-)
Weapons Fired	2	Good (+)	Bad (-)

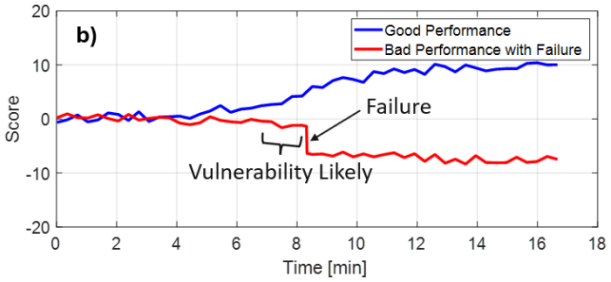

Figure 8. Sample Scoring Configuration and Resulting Scoring Plot. a) Values and weights are from blue's perspective. b) The gradient of the score curve can indicate a likely vulnerability.

To score the state vector, the analyst assigns value to each data feature within the kill-web. For example, consider Figure 8a which demonstrates scoring weights and direction for three features from Blue's perspective. 'Weapons Fired' represents a significantly important event and is weighted the highest as a result. If Blue fires a weapon, this is a positive value, whereas if Red fires the weapon, this results in negative value. Tasking of platforms is the second most important and is given half the weight of weapons firing. Finally, identifying a track is the least important of these three quantities and carries the smallest weight. The score weight is multiplied directly against the raw value of data to generate a score for that feature. To score the state, all feature scores are summed together. Through these weights, and through the features that are developed, the analyst defines what is important and configures the story that is represented by the graph. In this case, a Red fired weapon would most likely result in a sharp step down in score and is the key vulnerability. Finding moments just before this event may indicate where the kill-web can be improved.

The scoring method is applied to the test mission with the results shown in Figure 9. Currently, the left plot is generated automatically, but the understanding on the right is still determined through a manual investigation. In the Phase II program, additional analysis metrics are being designed to help an analyst quickly identify vulnerable moments and trace cause and effect around that vulnerability.

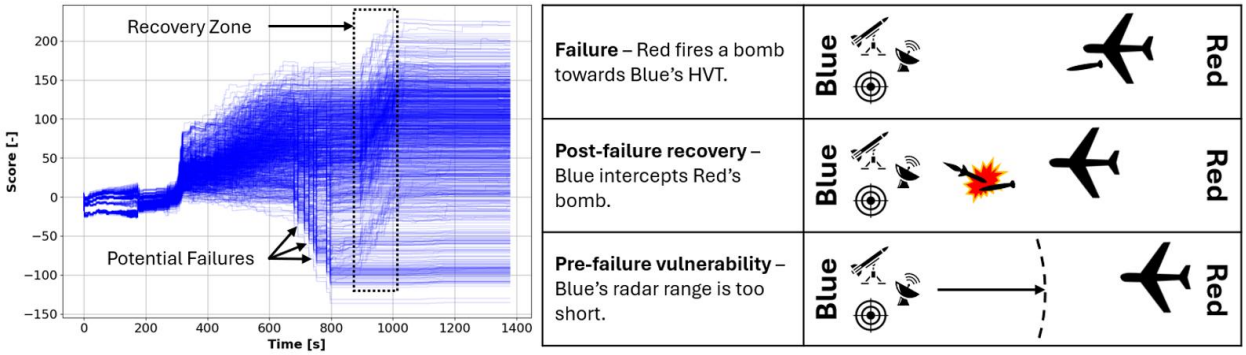

Figure 9. Scoring Analysis on the Test Mission. 1000 test simulations reveal potential failures and recovery. Through manual investigation, the reasons for failure, recovery, and the identified vulnerability are listed.

CONCLUSIONS

In this paper, Infinity presents a method to vectorize mission-level simulations in a step towards full Markovian evaluation of kill-webs. The state vector provides the means to mathematically connect webs of platforms based on an analyst's selection of mission-relevant features. Through this, the analysis traces cause and effect near vulnerable moments. Building the state vector for a kill-web scales exponentially with the number of platforms, quickly becoming computationally infeasible on today's high-performance computers. To address this, Infinity presents three main pillars to reduce complexity. First, feature simplicity means to leverage integer or binary features only. Second, feature compression outlines a method to translate continuous variables into the simplistic requirements of integer or binary. Third, merging and grouping of platforms to mimic situational awareness reduces the exponential scaling imposed by additional platforms on the state space. Leveraging the state vector, a sample investigation is performed on a test mission to identify failures, recovery methods, and the cause of those failures by scoring state vectors and observing mission performance over time. The work outlines the opportunity for mathematical representation of complex kill-web systems for higher fidelity technology comparisons at the mission level.

ACKNOWLEDGEMENTS

This work was supported by the US Air Force under contract FA875024CB064. The views expressed are those of the author and do not reflect the official policy or position at the US Air Force, Department of Defense, or the US Government. The author would also like to acknowledge his colleagues at Infinity Labs working towards the development of AASK: Bill Coughenour, Dan Huffman, and Dr. Josh Kogot. It is motivating and inspiring to work alongside a group dedicated to quality and putting this work into action.

REFERENCES

Bernal, I. (2020). Optimizing Engagement Simulations Through the Advanced Framework for Simulation, Integration, and Modeling (AFSIM) Software. *The Ohio State University M.S. Thesis*.

Billingsley, P. (1961). Statistical Methods in Markov Chains. *The annals of mathematical statistics*, 12-40.

DARPA. (2018). *DARPA Tiles Together a Vision of Mosaic Warfare.* Retrieved October 2023, from https://www.darpa.mil/work-with-us/darpa-tiles-together-a-vision-of-mosiac-warfare

Enders, J. (2023). Methodology for the Selection of Pivotal Features in AFSIM Meta-Markov Models. *Master's Thesis, The Ohio State University*.

Hartigan, J. A., & Wong, M. A. (1979). A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics), 28*(1), 100-108.

Hillstrom, D., Midlam-Mohler, S., Jankord, G., & Isaman, J. (2024). Markovian Analyses on Mission-Level Simulations with Many Platforms. *DAF M&S Summit 2024.* Houston.

Palm, E. A. (2023). An Approximate Dynamic Programming Approach for Solving an Air Combat Maneuvering Problem with Directed Energy Weapons. *Air Force Institute of Technology Thesis*.

Reinsel, D., Gantz, J., & Rydning, J. (2018). *The Digitization of the World from Edge to Core.* Framingham, MA: International Data Corporation.

Wang, K., Wang, W., Zhao, Y., Yuan, B., & Xiang, Z. (2023). Multisensor fault diagnosis via Markov chain and Evidence theory. *Engineering Applications of Artificial Intelligence, 126*. doi:https://doi.org/10.1016/j.engappai.2023.106851

West, T. D., & Birkmire, B. (2019). AFSIM: The Air Force Research Laboratory's Approach to Making M&S Ubiquitous in the Weapon System Concept Development Process. *Journal of Cyber Security and Information Systems*, 50-55.