Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

IMPLEMENTING EFFICIENT DYNAMIC THREAT AVOIDANCE ROUTING BASED ON DIJKSTRA'S SHORTEST PATH ALGORITHM IN THE ADVANCED FRAMEWORK FOR SIMULATION, INTEGRATION, AND MODELING (AFSIM)

Dante C. Reid
Lance E. Champagne
Nathan B. Gaw

Air Force Institute of Technology
Department of Operational Sciences
2950 Hobson Way
WPAFB, OH 45433-7765 USA

ABSTRACT

Simulating pre-planned routes and dynamic threat avoidance routing represents a significant problem for operations analysts. Without methods to create operationally valid routes through automation, the analyst is generally faced with hard coding individual routes for multiple aircraft over the entirety of the mission set. This research developed, implemented, and analyzed threat avoidance routing based on Dijkstra's algorithm for aircraft attempting to operate in an anti-access area denial (A2AD) environment capable of dynamically updating the mission route as new threat information is learned. A designed experiment was conducted to determine the impact of grid parameters on operational effectiveness metrics and computational costs. Statistical analysis results show that the proposed algorithm produced the best operational performance with grid spacing set to 50% of the smallest surface to air missile (SAM) threat radius without incurring prohibitive computational costs.

1 INTRODUCTION

Route planning for aircraft missions takes considerable time and is usually done days before missions are conducted. Rarely will a mission start with all the information on hand required to reduce risks from threats. Additionally, subsequent to initial mission planning, the location and number of threats could change rendering as new threats are discovered through Airborne Intelligence, Surveillance, and Reconnaissance (ISR) or by onboard sensors during the mission itself. Calculating new routes in flight presents a daunting challenge to aircrew that are already task-saturated.

The 2018 National Defense Strategy discusses delivering performance at the speed of relevance, innovation, and streamlining rapid, iterative approaches from development to fielding (Mattis 2018). Innovation drives developments enhancing tactical advantage, which can help with the development and fielding of these systems as well (Trevithick 2019). As a result, modern military aircraft make use of sensor fusion to modify routes in-flight to mitigate the threat from opposition systems. The sophistication of these systems allows the aircrew to minimize exposure to opposition sensors and change routing dynamically as new threats become evident.

Simulating pre-planned routes, generally, and threat avoidance routing, particularly, represents a significant problem for operations analysts. Without methods to create operationally valid routes through automation, the analyst is generally faced with hard coding individual routes for multiple aircraft over the entirety of the mission set. Therefore, developing a routing algorithm capable of computing the path of

least risk through a contested area with the capability of dynamically updating the route based on new threat information is a critical tool for producing relevant simulation results.

This research investigates implementation of a routing algorithm based on Dijkstra's shortest path algorithm developed for combat simulations in the Advanced Framework for Simulation, Integration, and Modeling (AFSIM). The algorithm initially calculates a route minimizing the threat exposure to the aircraft and dynamically updates the route based on new threat information during mission execution.

The remainder of this article is organized as follows. Section 2 is the literature review of A2AD, Combat Modeling, Dijkstra's Algorithm, Military Threat Routing, and Design of Experiments. Section 3 is the methodology that went into the study and its framework for the AFSIM model implementation with internal routing algorithms, simulated A2AD scenario, Dijkstra Routing Tool, Design of Experiments (DOE), and measures of effectiveness (MOEs). Section 4 includes the results and analysis of the simulation runs. Section 5 contains the conclusions and recommendations drawn from this research and considerations for future research.

2 BACKGROUND AND RELEVANT LITERATURE

In this section, the following topics are covered: A2AD, Combat Modeling, Dijkstra's Algorithm, Military Threat Routing, and DOE.

2.1 Anti-Access Area Denial

A2AD is not a new threat but is becoming more prevalent as a response to Western force projection, precision strike, and highly-networked Command and Control (C2) capabilities (Schmidt 2016). Anti-Access refers to strategies preventing the enemy from entering the theater of operations and Area Denial refers to the operations that restrict the freedom of maneuver once a threat has entered the theater of operations (Krepinevich et al. 2003). Specifically, near-peer countries have implemented the A2AD strategy for the sea domain in the Asia-Pacific region (Permal 2014). However, A2AD is not limited to near-peer adversaries, as it is a cost-effective means, regardless of overall military strength, to defend against an opponent with superior capabilities. A2AD has different strategies employed for sea, land, air, and cyberspace domains. This research focused on only the air domain and explores dynamic routing with a combat simulation for a penetration asset against generic SAMs with independent radar systems deployed in an A2AD strategy.

2.2 Combat Modeling

Combat modeling is an effective means of simulating military operations from large scale low-resolution aggregated models for strategic wargaming to high-resolution models with individual platforms, since military exercises are costly, both in time and money. The resolution and aggregation of a combat model depends on the objective, such as theater-level, multiservice invasion campaigns or individual platform testing. The difference in resolution affects the insights obtained and can be seen in Zeigler's research where aggregation does not produce the same results as a high-resolution model (Zeigler 2017).

The goal of combat modeling is to make the simulation realistic enough to produce meaningful insights. Agent-Based Modeling and Simulation (ABMS) is an effective tool used to produce results that are a direct representation of the real world (Macal 2016). The results from ABMS provide insight that could demonstrate trends of actual combat behaviors (Hill et al. 2004). This research uses a high-resolution model that is stochastic, dynamic, and continuous. Additionally, there are static and discrete elements in the model, such as SAM locations and threat identification intervals.

2.3 Military Threat Routing

There are numerous studies on route planning algorithms for both military and commercial applications. Many DoD-related operational route planning algorithms are classified or proprietary, and as they are not

real-time or during flight (Szczerba et al. 2000), they are outside the scope of this research. Likewise, routing algorithms have been a focused research area in the literature. O'Rourke et al. investigated dynamic vehicle routing for Unmanned Aerial Vehicles (UAV) using Java-encoded metaheuristics, which yielded close to optimal solutions within reasonable computing times (O'Rourke et al. 2001). The dynamic routing portion of their research centered on taking advantage of pop-up targets of opportunity, not threat avoidance.

Szczerba et al. implemented real-time route planning using a novel variation of the A* search algorithm, which they called Sparse A* Search (SAS) (Szczerba et al. 2000). SAS was able to work in real-time with dynamic threat locations and identify threats not visible to an aircraft flying at the altitude specified for a segment of the route. The authors suggested enhancements to encompass higher dimensional environments, multiple/dynamic targets, and additional constraints.

Selecting the appropriate route to avoid obstacles is crucial to this research. Baxter and Warren investigated route selection in barrier avoidance and proposed utilizing waypoints at the end of barriers. The same thought process could apply to threat avoidance with the "barriers" defined by edges of SAM threat rings (Baxter and Warren 2020). These waypoints would minimize the cumulative amount of turning for the planned path, which could be beneficial for the penetration assets. To apply this dynamically, waypoints are recalculated from the initial route depending on changes to previously unidentified enemy SAMs. However, this approach is not applicable for an A2AD environment in which the area is saturated with SAMs, leaving no safe path to the target.

2.4 Djikstra's Shortest Path Algorithm

Dijkstra's algorithm finds the shortest path between two vertices (also referred to as nodes or points). Proposed by Edsger W. Dijkstra, Dijkstra's algorithm does not require data for all branches simultaneously and is computationally more efficient than many other competing algorithms (Dijkstra 2022), making it ideal for this research. Dijkstra's algorithm is a greedy algorithm with a time complexity of $O(E \log V)$, assuming that the graph is fully-connected and where E is the total number of edges and V is the number of vertices (Danziger 2010).

Dijkstra's algorithm only requires two steps but the vertices and edges (also referred to as roads or branches) are subdivided into three sets each (Dijkstra 2022). Edges are the connections between vertices that are usually annotated with a length. The algorithm subdivides the vertices accordingly:

Set A. the vertices whose minimum length from the starting vertex is known;
Set B. the vertices where one of them will be the next vertex added to set A;
Set C. the rest of the vertices.

The edges are subdivided accordingly:

Set I. the edges whose minimal paths from the starting vertex to the vertices in set A;
Set II. the edges where one of them will be the next edge added to set I;
Set III. the rest of the edges who have either been rejected or not yet considered.

The algorithm initializes with all vertices in set A and all edges in set III. The starting vertex is moved into set A, and algorithm proceeds to the first step below.

The first step considers all edges connected to the last vertex transferred to set A to the other vertices in sets B or C. The shortest edge connecting the latest node in set A to a member of set B is identified, and set II is updated accordingly.

The second step considers all the members of set B that were updated in the first step. The vertex with the shortest distance from the starting vertex in set B is moved to set A and the corresponding edge is moved from set II to set I. The algorithm then returns to the first step until the destination vertex is moved to set A, resulting in a shortest path solution.

Brown showed that Dijkstra's algorithm is a simple and efficient algorithm appropriate for UAV routing (Brown 2001).

3 METHODOLOGY

This section discusses the methodology used to explore the effectiveness of the Dijkstra's shortest path algorithm in a combat simulation for threat avoidance routing. The following topics are covered: AFSIM, Operational Scenario, Dijkstra's Routing Tool, DOE construct, and MOEs.

3.1 AFSIM

AFSIM is an agent based combat simulation environment written in C++ for engagement and mission level combat analysis. Originally created by Boeing, AFSIM is now owned by the US government and is under the management of Air Force Research Laboratory (AFRL). Through AFSIM, new system concepts and designs can be tested ranging from "weapon kinematics, sensor systems, electronic warfare systems, communication networks, advanced tracking, correlation, and fusion algorithms, and automated tactics and battle management software" (Clive et al. 2015).

AFSIM has default options for platforms and their components as well as scripts used to define agent behaviors outside the default options. The Dijkstra's Routing Tool, detailed in section 3.3, was coded as an agent scripted behavior. In a comparison between default routing options written as add-ins in the baseline C++, the script implementation would disadvantage the proposed routing algorithm in computational speed. However, this comparison is outside the scope of this investigation. This research utilized AFSIM version 2.8.3.

3.2 Simulated Operational Scenario

The scenario is custom built and extends the heavy scenario used in (Tucker 2021). The enemy SAMs are defending an arbitrary geographical region in southwest Ohio. They are radar-based and act autonomously.

The penetration asset's mission is to travel from a rally point, (38°55'N, 87°2'W), to the target located at (39°44'N, 83°19'W), a distance of 180.6 nautical miles (NM). The penetration asset has a speed of 480 knots at 10,000 ft altitude with a maximum radial acceleration of 6G. While the simulation does account for geography in ground-to-air engagements, given the terrain and attack altitudes chosen, terrain is not a factor in SAM effectiveness. Additionally, the aircraft is set to be "indestructible," allowing it to potentially be hit multiple times in route to the target. This allows for some indication of the level of lethal threat exposure certain routes may give the penetration asset. The simulation ends after two hours of simulation time.

The enemy defense is composed of three types of generic SAMs - small, medium, and large - differing only in their radar and missile system capabilities. Large SAMs have a maximum detection range of 60 NM; medium SAMs have a maximum detection range of 40 NM; and small SAMs have a maximum detection range of 20 NM. All SAM types use a single radar sensor that operate at ten (10) GHz frequency with a frame time of two (2) seconds.

The SAM firing behavior is script-based. Radar tracking and hit are stochastic and subject to random draws. The kinetic (lethal) ranges for the SAM types are the same as their detection range. If the penetration asset is being tracked by the SAM then the SAM will fire a single missile at the penetration asset at a maximum rate of one missile every two minutes. All SAMs have the same probability of hitting the penetration asset based on the range the of the penetration asset to the SAM firing. If the penetration asset is between $\frac{2}{3}$ and the maximum range then there is a 30% chance that the missile will hit the penetration asset. If the penetration asset is between half and $\frac{2}{3}$ of the maximum range then there is a 50% chance that the missile will hit the penetration asset. If the penetration asset is within $\frac{1}{2}$ of the maximum range then there is an 80% chance that the missile will hit the penetration asset.

The A2AD scenario has eleven (11) SAMs assigned between four (4) different groups as pictured in Figure 1. Each SAM group is independent of the others, but all SAMs within a group are created at the same simulation time. The varying creation times allow for different levels of threat to become visible (discoverable) to the penetration asset as the mission progresses.

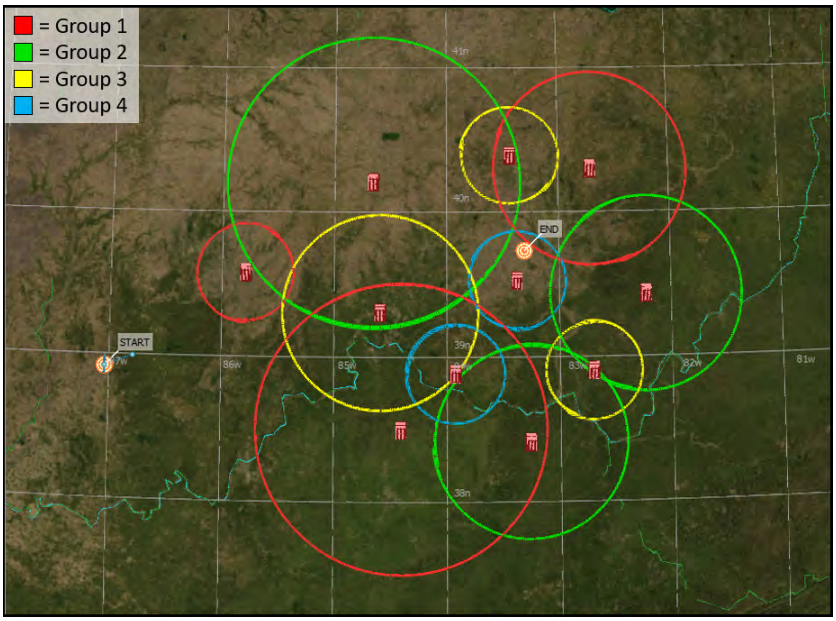

Figure 1: SAM Groupings.

During the simulation run, these groups will have the creation time rules of becoming visible at the beginning of the simulation run, the 150^{th} second, the 510^{th} second, or 750^{th} second of simulation time, respectively. The SAMs present at the time the simulation begins represent known threats identified with advanced intelligence. SAMs that appear subsequently represent SAMs that were identified by either airborne ISR or the penetration asset itself during the mission. The creation times were established based on the penetration asset's location when using a straight line route to the target to ensure all SAMs were discoverable before the penetration asset reached the target, while allowing sufficient time to adjust its route based on its updated site picture.

3.3 Dijkstra's Routing Tool

The implementation of Dijkstra's algorithm was originally developed in (Tucker 2021) and extended in this research to include dynamic rerouting. The Dijkstra Routing Tool uses the AFSIM scripting language and is a modified version of the Dijkstra shortest path algorithm. Traditionally Dijkstra's algorithm finds the shortest path between two points in a network but the waypoints in this network are all equally spaced. Since this research looks for the safest path through enemy territory, the waypoints themselves have scores that determine how much of a threat that particular waypoint is. The Dijkstra Routing Tool scores the waypoints based on their proximity to SAMs. The algorithm then finds a route to the target with the lowest cumulative score, which theoretically translates to the least threat exposure.

The Dijkstra Routing Tool starts by generating a grid of waypoints from the current location to the ending location. The network is established as a rectangular area specified by a fraction of the straight-line distance remaining to the target parallel and perpendicular to the line between the current and ending locations. The network nodes are generated from the current location throughout the rectangular area according to the grid spacing. The fractions determining the network area and the grid spacing are user defined as simulation inputs. As an example, if the grid length percentage is 125% with a grid width

percentage of 75% and the distance between the starting and ending location is 200 kilometers, then the grid will be 250 kilometers long by 150 kilometers wide. Network nodes will be spaced according to the grid spacing parameter. If the grid spacing is 25 kilometers then the previous example would be 10 waypoints long by 6 waypoints wide for a total of 60 waypoints that need to be evaluated. The waypoints are fully connected to their adjacent waypoints both on the primary axes and diagonally, meaning waypoints not on an edge have 8 arcs to adjoining waypoints (Tucker 2021).

After the Dijkstra Routing Tool generates all the waypoints, the waypoints are then evaluated for a threat score based on their proximity to all the SAMs currently generated in the simulation scenario. The threat cost of a waypoint has a default value of one (1). Additional costs are assessed based on a linear threat cost function of a waypoint's distance to every SAM. Within 50% of the maximum range of a SAM, the threat score is an additional 100. Outside of 50% maximum range, the added threat score linearly decreases to zero (0) when outside of the maximum range. For example, the medium SAMs would have a threat score of 100 for any waypoint within 20 NM of the SAM, a threat score of 50 for any waypoint that is 30 NM of the SAM, and a threat score of zero (0) for any waypoint that is outside of 40 NM of the SAM. Each SAM has the potential to add threat score to a node. Therefore, it is possible for waypoints to have threat scores higher than 100 in instances where a waypoint is within the threat ring of multiple SAMs (Tucker 2021).

Finally the Dijkstra Routing Tool looks for the path through enemy territory that incurs the least total threat cost. An example of the Dijkstra Routing Tool's outputs can be seen in Figure 2 that shows a heat map of the waypoints threat score and the generated route.

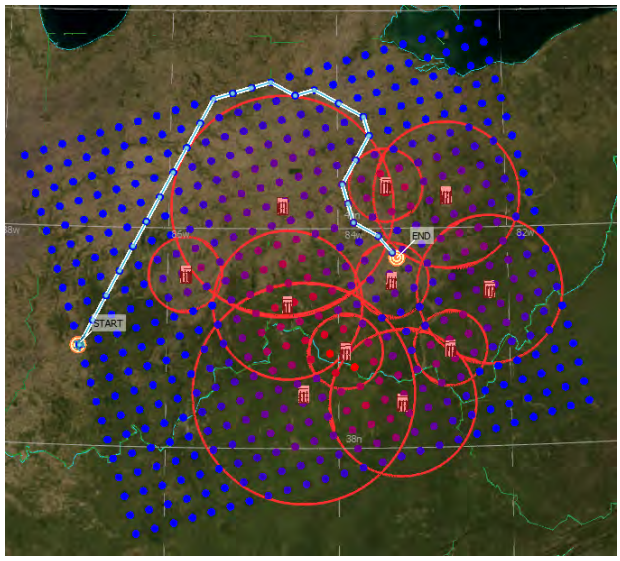

Figure 2: Dijkstra Route Generation Example.

3.4 Designed Experiment

The designed experiment was implemented through Harness for Adaptive Learning (HAL). HAL is an adaptive design-and-learning tool for efficient sampling. Written in Python, HAL integrates with AFSIM to generate space-filling and uniform experimental designs using a maximum projection design (Tournay 2022). A maximum projection experiment design was used for this research, which has demonstrated better space-filling properties than a maximum projection Latin hypercube design, maximin Latin hypercube design, uniform design, and generalized maximin Latin hypercube when there are less than nine (9) factors (Joseph et al. 2015).

The factors for the experiment were number of SAM waves (0, 1, 2, and 3), number of total SAMs at the end of the simulation (3, 5, 6, 8, 9, 11), and algorithm grid spacing (ranging from 5,000 to 52,000 meters, determined by the maximum projection design). The experiment consisted of 1,028 design points with 30 replications each, resulting in 30,840 experiment trials.

The number of waves of SAMs and the total number of SAMs were selected to test the threat avoidance effectiveness over a variety of challenging scenarios requiring different numbers of route recalculations.

The grid spacing bounds were selected based on the limits of aircraft maneuverability and the lethal radii of the SAM threats. The lower bound is set such that the kinematic performance of the aircraft would permit the aircraft to travel between neighboring nodes. The upper bound was determined by the threat ring of the small SAM. Based on the formula of a circle shown in Equation 1 with R being the radius of the smallest SAM threat ring and X being the grid spacing of the waypoints, one can determine the largest grid spacing that would guarantee at least one node intersects all SAMs in the flight area.

$$R^2 = (\frac{X}{2})^2 + (\frac{X}{2})^2 \tag{1}$$

From this, calculations show that grid sizes larger than 52,374.56 meters had the potential to allow nodes to fall completely outside of the threat area covered by a small SAM. Therefore, the upper bound was set to 52,000 meters to ensure that at least one waypoint would fall within the smallest SAM threat ring if present in the potential aircraft routes to the target.

3.5 Measures of Effectiveness

Through analysis of AFSIM output, we use the following MOEs:

3.5.1 MOE 1: Number of Shots Taken at the Penetration Asset

This is an operational MOE, and it quantifies the total number of times enemy SAMs fired at the penetration asset. This MOE acts as a survivability metric for the penetration asset.

3.5.2 MOE 2: Number of Hits on the Penetration Asset

This is an operational MOE, and it quantifies the total number of times the enemy SAMs successfully hit the penetration asset. This MOE acts as a survivability metric for the penetration asset.

3.5.3 MOE 3: Total Time the Penetration Asset was Tracked by the Enemy

This is an operational MOE and it quantifies the amount of simulation time the penetration asset was within firing range of an enemy SAM. Since the SAMs are independent of one another, it is possible for 'Total Time Penetration Asset was Tracked by the Enemy' to be longer than 'Mission Duration Time' because multiple SAMs may add to the tracking time simultaneously. This MOE acts as a vulnerability metric for the penetration asset.

3.5.4 MOE 4: Mission Duration Time

This is an operational MOE and it quantifies how long the penetration asset took to get from the start point to the end point in simulation time seconds. This MOE acts as both a vulnerability and lethality metric, depending on the analysis focus.

3.5.5 MOE 5: Computational Processing Time

This is an computational MOE and it quantifies the amount of time the computer took to initialize and execute the AFSIM simulation in real-world seconds. The start time is subtracted from the end time to get the amount of time in real world seconds the computer took to process the simulation. This MOE acts as

a indication of, or proxy for, the computational complexity for the dynamic route planning for its various simulation settings.

4 RESULTS

The results from the full DOE address broader questions from (Reid 2023). For the purposes of this analysis, design points were binned according to the grid spacing factor. Binning allows confidence interval analysis to meet appropriate underlying assumptions of normality and variance. Results from the experiment were computed as 95% confidence intervals for the purposes of statistical comparison. Each of the MOEs are detailed in the following sections.

A note about interaction between factors: Some of the interactions were statistically significant but were practically negligible. For example, interactions account for 2.52% of the total effects in the case of MOE 5, which was the largest contribution of any MOE investigated.

4.1 MOEs 1 and 2: Number of Shots and Hits Against the Penetration Asset

Figure 3 shows the mean number of shots taken at the penetration asset with its 95% confidence intervals for the different grid spacing. The smaller the mean, the better chance the penetration asset has for survivability. Generally, as the grid spacing increases, the number of shots taken at the penetration asset also increases. By increasing the grid spacing, fewer routes are available to the penetration asset, and the aircraft is less able to find routes without threat exposure. As a result, the aircraft faces more SAM shots. Of note, grid spacing settings between 13K-21K meters and 21K-29K meters are not statistically different. However, the computational time associated with the smallest grid spacings are statistically (and practically) greater (discussed in 4.4). Therefore, the 21K-29K meter grid spacing for Dijkstra's algorithm performed better operationally without incurring undue computational penalty. This grid spacing is roughly 50% of radius associated with the smallest SAM.

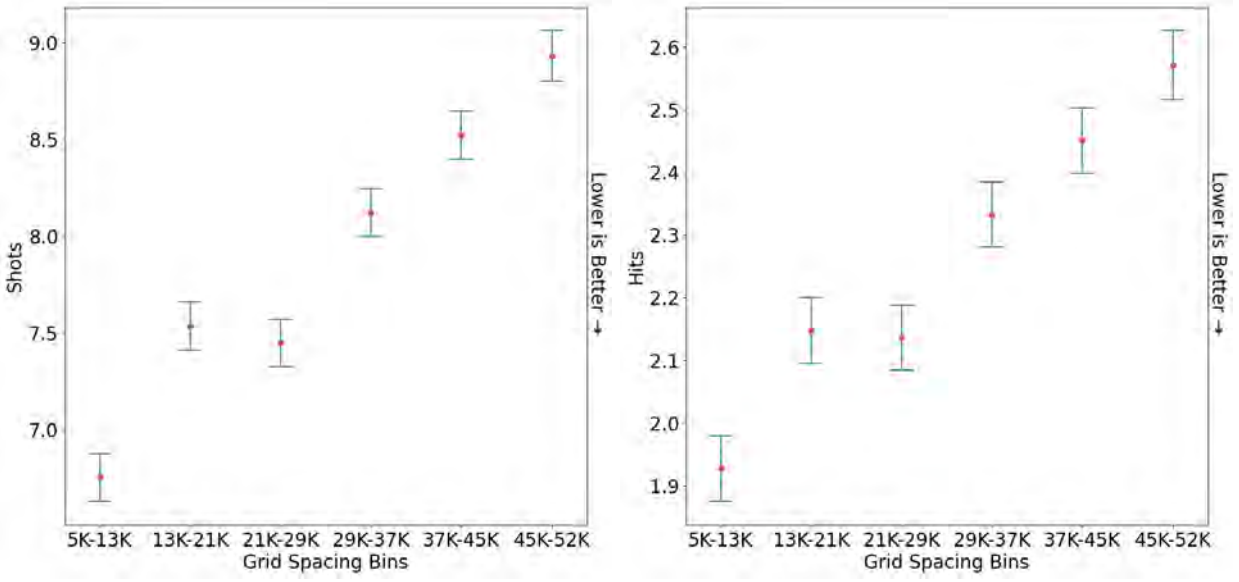

Figure 3: Number of Shots (left) and Hits (right) Against the Penetration Asset vs. Grid Spacing.

MOE 2 is very similar to MOE 1. Intuitively, more SAM shots would lead to more hits, although the distance from aircraft to SAM affects the SAM lethality. The smaller the mean, the better chance the penetration asset has for survivability. As with MOE 1, as the grid spacing increases the number of hits

on the penetration asset also increases. Additionally, the grid spacing settings between 13K-21K meters and 21K-29K meters are not statistically different.

4.2 MOE 3: Total Time the Penetration Asset was Tracked by the Enemy

Figure 4 shows the mean total time the penetration asset was tracked by the enemy with its 95% confidence intervals for the different grid spacing. The smaller the mean, the better chance the penetration asset has for survivability. This MOE does not show statistical differences between design setting, so there are no conclusions that may be drawn from the graph.

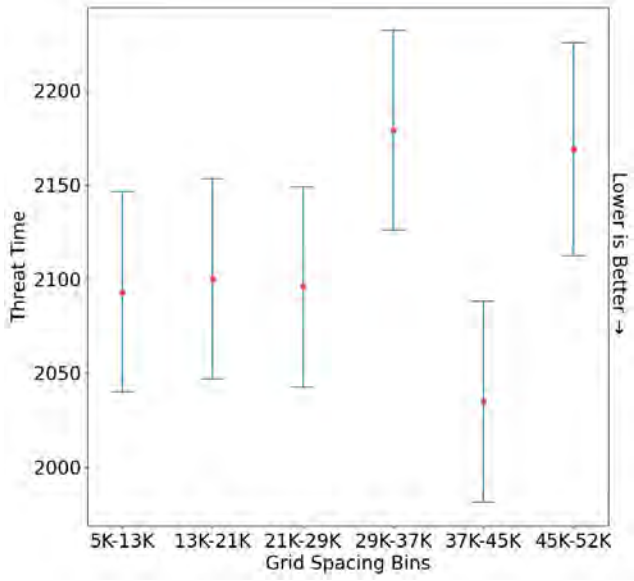

Figure 4: Total Time the Penetration Asset was Tracked vs Grid Spacing.

4.3 MOE 4: Mission Duration Time

Figure 5 shows the mean mission duration time with its 95% confidence intervals for the different grid spacing. The smaller the mean, the quicker the penetration asset reached the target. Generally, as the grid spacing increases the mission duration time decreases with a spike at 29K-37K meters and negative spike at 37K-45K meters. The larger grid sizes reduce the routing options available to the aircraft, and as a result, it takes a more direct route to the target. Further analysis is required to determine the possible reason for the statistically shorter mission duration seen with grid spacing between 37K-45K meters.

4.4 MOE 5: Computational Processing Time

The total experiment took 40,609.36 seconds (11.28 hrs) to run on our system. Of this, 32,917.33 seconds (9.14 hrs) constituted scenario computational processing time. Figure 6 shows the mean computational processing time with its 95% confidence intervals for the different grid spacing. The smaller the mean, the less computational penalty imposed by the dynamic routing algorithm. As the grid spacing increases, the computational processing time decreases and is significant across the design settings. In fact, the processing time shows an exponential curve in computation time growth as the distance between nodes decreases. Visual inspection reveals that the "knee in the curve" occurs in the grid spacing between 13K-29K meters. Smaller grid spacing tends to incur a relatively heavy penalty for invoking the dynamic routing algorithm.

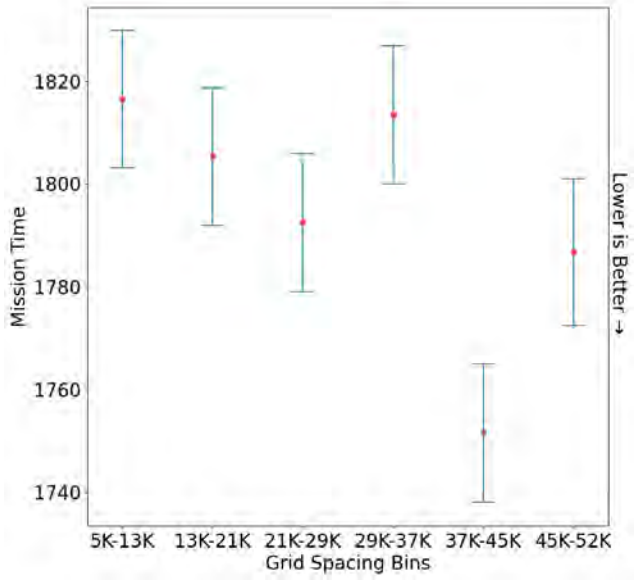

Figure 5: Mission Duration Time vs Grid Spacing.

4.5 Insights

The experiment showed that as grid spacing increased so did the number of shots taken at the penetration asset (MOE 1) and number of hits on the penetration asset (MOE 2) with the difference between the 21K-29K meters and 13K-21K meters spacing not being statistically significant. MOE 3 does not provide additional insight without additional experimentation and analysis. The 37K-45K meter grid spacing outperformed the other grid spacing bins with mission duration time (MOE 4). Moreover, as the grid spacing increased, the computational processing time (MOE 5) decreased in an exponential decay. Overall, the 21K-29K meter grid spacing for Dijkstra's algorithm performed better operationally without incurring undue computational penalty. This grid spacing is roughly 50% of radius associated with the small SAM.

5 CONCLUSIONS AND FUTURE RESEARCH

As adversarial forces make technological advances, the ability to analyze capabilities and strategies outside of actual combat through combat modeling is a critical tool. Dynamic routing of agents in combat models is necessary to account for updated intelligence after mission start. Dynamic routing approaches for aircraft must be able to calculate quickly the safest route for the pilot and in minimal time to ensure the aircraft survives to complete the mission. This research implemented a dynamic threat avoidance routing algorithm based on Dijkstra's shortest path algorithm to provide this capability within the AFSIM environment.

The results from the simulation experiment showed there are computational and operational tradeoffs in the grid spacing for the dynamic routing. Although the grid spacing above 29,000 meters had low computational processing time, it also resulted in an increase in number of shots taken at the penetration asset, number of hits on the penetration asset, and total time the penetration asset was tracked by the enemy. The processing times for the grid spacing below 21,000 meters were large with marginal improvements in the operational MOEs.

Results where the grid spacing was 21K-29K meters, which is roughly 50% of the radius associated with the smallest threat ring, appeared to be the best grid spacing. In these cases, computational processing times were improved and statistically significant from smaller grid spacings. Additionally, the computational times were only marginally increased from those from larger grid spacing. Thus, the grid spacing for Dijkstra's algorithm that performed better operationally without sacrificing too much computationally were with nodes spaced between 21K-29K meters.

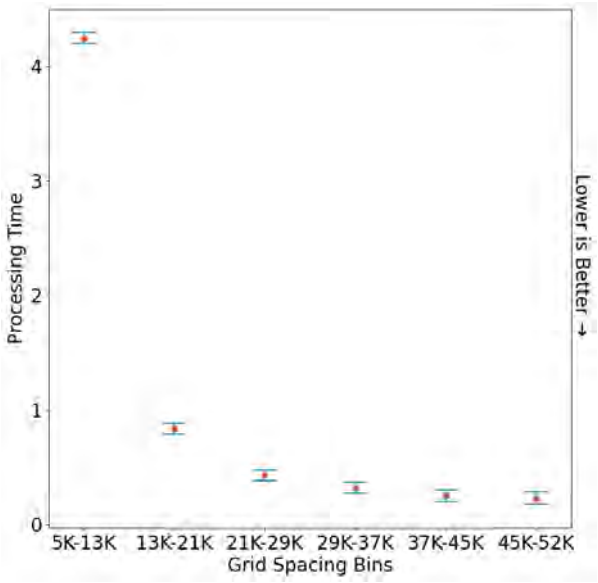

Figure 6: Computational Processing Time vs Grid Spacing.

There are myriad opportunities to extend this research. Additional experimentation using different cost functions for the algorithm may provide increased operational benefits. While the results found that the "sweet spot" for grid spacing was near 50% of the smallest lethal radius, more experimentation under different scenarios is needed to determine if this could be generalized as a rule of thumb or is a happenstance of this particular scenario. Moreover, the threat routing considered two-dimensional paths at constant altitude. There may operational benefits in computing a three-dimensional path that would exceed the additional computational complexity levied by the additional dimension. Additionally, this would allow better consideration of terrain in the route planning. Finally, a variant of Dijkstra's algorithm was selected for its computational efficiency and ease of implementation. However, there are many other efficient routing algorithms that could be implemented. An example would be Particle Swarm Optimization (PSO) that has shown the potential to outperform Dijkstra's Algorithm (Kang et al. 2008).

REFERENCES

Baxter, B. A., and W. H. Warren. 2020. "Route Selection in Barrier Avoidance". *Gait & Posture* 80:192–198.

Brown, D. T. 2001. "Routing Unmanned Aerial Vehicles While Considering General Restricted Operating Zones". Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH.

Clive, P. D., J. A. Johnson, M. J. Moss, J. M. Zeh, B. M. Birkmire, and D. D. Hodson. 2015. "Advanced Framework for Simulation, Integration and Modeling (AFSIM) (Case Number: 88ABW-2015-2258)". In *Proceedings of the 2015 International Conference on Scientific Computing*, 73. The World Congress in Computer Science, Computer Engineering and Applied Computing.

Danziger, P. 2010. "Big O Notation". *http://www.scs.ryerson.ca/˜mth110/Handouts/PD/bigO.pdf, Accessed: April 2022*.

Dijkstra, E. W. 2022. "A Note on Two Problems in Connexion with Graphs". In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 287–290. New York, NY: Association for Computing Macinery.

Hill, R. R., L. E. Champagne, and J. C. Price. 2004. "Using Agent-Based Simulation and Game Theory to Examine the WWII Bay of Biscay U-Boat Campaign". *The Journal of Defense Modeling and Simulation* 1(2):99–109.

Joseph, V. R., E. Gul, and S. Ba. 2015. "Maximum Projection Designs for Computer Experiments". *Biometrika* 102(2):371–380.

Kang, H. I., B. Lee, and K. Kim. 2008. "Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm". In *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, Volume 2, 1002–1004. Institute of Electrical and Electronics Engineers.

Krepinevich, A. F., B. D. Watts, and R. O. Work. 2003. *Meeting the Anti-Access and Area Denial Challenge*. Washington, DC: Center for Strategic and Budgetary Assessments.

Macal, C. M. 2016. "Everything You Need to Know about Agent-Based Modelling and Simulation". *Journal of Simulation* 10(2):144–156.

Mattis, J. 2018. "Summary of the 2018 National Defense Strategy of the United States of America". Technical report, Department of Defense, Washington, DC, United States.

O'Rourke, K. P., W. B. Carlton, T. G. Bailey, and R. R. Hill. 2001. "Dynamic Routing of Unmanned Aerial Vehicles Using Reactive Tabu Search". *Military Operations Research* 6:5–30.

Permal, S. 2014. "China's Military Capability and Anti-Access Area-Denial Operations". *Maritime Affairs: Journal of the National Maritime Foundation of India* 10(2):16–32.

Reid, D. 2023. "Simulation and Analysis of Dynamic Threat Avoidance Routing in an Anti-Access Area Denial (A2AD) Environment". Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH.

Schmidt, A. 2016. "Countering Anti-Access/Area Denial Future Capability Requirements in NATO.". *Transforming Joint Air Power* 23:69–75.

Szczerba, R. J., P. Galkowski, I. S. Glicktein, and N. Ternullo. 2000. "Robust Algorithm for Real-Time Route Planning". *IEEE Transactions on Aerospace and Electronic Systems* 36(3):869–878.

Tournay, R. 2022. "Harness for Adaptive Learning". Technical report, Simulation Technology Interchange Meeting 2022. Defense Innovation Marketplace.

Trevithick, J. 2019. "USAF Wants to Network Its Precision Munitions Together into a 'Golden Horde' Swarm". *https://www.thedrive.com/the-war-zone/28706/usaf-wants-to-network-its-precision-munitions-together-into-a-golden-horde-swarm, Accessed: April 2022*.

Tucker, W. 2021. "Simulation and Analysis of Dijkstra's Routing Algorithm for A2AD Using AFSIM". Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, OH.

Zeigler, B. P. 2017. "Constructing and Evaluating Multi-Resolution Model Pairs: an Attrition Modeling Example". *The Journal of Defense Modeling and Simulation* 14(4):427–437.

AUTHOR BIOGRAPHIES

DANTE C. REID is an operations research analyst for the United States Air Force and a graduate student in operations research at the Department of Operational Sciences at the Air Force Institute of Technology. He has a BA in Economics from Gonzaga University, MS in Operations Research from the Air Force Institute of Technology, and MAML from Webster University. His email address is dante.c.reid.mil@mail.mil

LANCE E. CHAMPAGNE is an Assistant Professor of Operations Research and Chair of the Modeling, Simulation, and Analysis Graduate Certificate Program at the Air Force Institute of Technology. His research interests include simulation of autonomous system behavior, agent-based combat simulation, and neural network design for image/video classification. He is a member of the Cincinnati/Dayton Chapter of the Institute for Operations Research and Management Sciences (INFORMS) and the Military Operations research Society (MORS). His email address is lance.champagne@afit.edu

NATHAN B. GAW received B.S.E. and M.S. degrees in biomedical engineering and a Ph.D. degree in industrial engineering from Arizona State University (ASU), Tempe, AZ, USA, in 2013, 2014, and 2019, respectively. He is currently an Assistant Professor with the Department of Operational Sciences, Air Force Institute of Technology, Wright-Patterson AFB, OH, USA. His research interests include multimodality fusion and semi-supervised learning in telemonitoring, military applications, and healthcare imaging. He is a member of INFORMS, IISE and IEEE. His email address is nathan.gaw@afit.edu