EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

Extending JASPER for Jammer-Angle Allocation Evaluation: Testing Algorithm Robustness Under Jammer Failures and Varying Resource Conditions

Ahmet Burak Baraklı[1,3,*], Okan Topçu[2], Halit Oğuztüzün[1]

[1]Middle East Technical University, Ankara, Türkiye
[2]Middle East Technical University, Northern Cyprus Campus, 99738, Kalkanlı, Güzelyurt, Mersin 10, Türkiye
[3]Aselsan Inc., Ankara, Türkiye

Abstract

JASPER is a scenario-driven simulation testbed for analyzing jammer-angle allocation (JAA) strategies in anti-UAV defense systems. It provides a modular simulation environment where users can run controlled experiments that reflect realistic anti-UAV scenarios. In this extended version, we introduce a new failure mechanism that allows jammers to fail dynamically during simulation. This addition enables the testing of algorithmic robustness under partial system degradation and is showcased through a newly designed case study focused on how algorithms recover and adapt after resource loss. We also conduct a sensitivity analysis by varying the number of jammers, exploring trade-offs between coverage, swivel workload, and algorithm runtime. Beyond enabling performance comparisons under realistic constraints, these experiments also reinforce the reliability and adaptability of the JASPER platform. Together, these additions make JASPER a more expressive and fault-aware testbed for evaluating directional jamming strategies under realistic conditions.

Received on 10 December 2024; accepted on 02 July 2025; published on 18 July 2025

Keywords: Simulation-based test and evaluation, Agent-based simulation, Jammer allocation algorithms, Threat evaluation and weapon allocation, Anti-drone systems

doi:10.4108/eetiot.9702

1. Introduction

With the development of technology in today's world, unmanned aerial vehicles (UAVs) are extensively used as weaponry, which has triggered the development of countermeasures. Anti-drone systems play an important role in protecting critical areas and facilities. These systems provide functions such as detecting, tracking, identifying, interfering, and assessing threats specifically against UAVs, usually in addition to traditional air defense systems [1, 2].

Detection and tracking of UAVs is accomplished through various sensors such as radars and camera systems. The detection phase is the primary step to ensure the effectiveness of anti-drone systems because preventive measures cannot be taken against undetected UAVs. Then detected trace is identified and a threat importance is assigned. As a result of this assignment, if it is deemed that the trace is dangerous and countermeasure is required, then the interfering starts. The ways to interfere with UAVs can be divided into two: hard kill and soft kill. Hard kill methods, such as laser weapons or missile systems, aim to physically destroy UAVs. In contrast, soft kill methods focus on neutralizing UAV activities without physical destruction, using techniques like signal jamming and frequency jamming. Jammers, a popular soft kill approach, disrupt the control of UAVs by interfering with their communication signals, effectively rendering them inoperable. Omni-directional jammers are commonly used for this purpose; however, in scenarios where surrounding signals (e.g., those used for daily communication) must not be disrupted, directional jammers that

*Corresponding author. Email: ahmet.barakli@metu.edu.tr

target specific angles are preferred [3]. Anti-drone systems can operate either under human supervision or autonomously. In autonomous mode, algorithms analyze and track information from sensors, calculate threat priorities, and execute necessary actions through jammers. This decision-making process needs to be completed rapidly because the acceptable response time against incoming threats is generally restricted to a few seconds [4, 5].

Given the increasing complexity and speed of UAV threats, developing effective algorithms for threat evaluation and weapon allocation is crucial. Various algorithms have been proposed, each with its strengths and weaknesses in terms of prioritization, computational efficiency, and adaptability to different types of threats and environments. To determine the most effective algorithms and tune their parameters, comprehensive testing under a wide range of scenarios is essential. However, real-world testing of these algorithms is costly and hard to reproduce. Thus, a simulation environment that can test and evaluate jammer-angle allocation algorithms to simulate different test scenarios and obtain the relevant metrics is needed. In response to this need, this study proposes a simulation tool named JASPER (Jammer Allocation Simulation Platform for Evaluation and Reporting). JASPER is a modular, agent-based environment for testing and evaluating the performance of various jammer-angle allocation algorithms, and it is based on the MASON framework [6]. It offers a customizable simulation for analysts and planners to analyze different key performance indicators (KPIs) and refine algorithmic tactics.

The primary contribution of JASPER is its modularity and adaptability, which let users customize the simulation environment to fit their unique requirements and scenarios. Through the integration of multiple algorithms, test cases, and reporting modules, JASPER offers an adaptable environment for the examination and comparison of different jammer-angle allocation techniques. This adaptability is especially useful for research and development contexts, where it allows for the quick prototyping, testing, and optimization of new algorithms against a variety of UAV threat scenarios.

Furthermore, JASPER's capability to integrate different reporter modules that record various KPIs, such as coverage percentages, algorithm run times, and angle adjustments, enables the evaluation of various results, supporting analysts and planners in determining the advantages and disadvantages of each tactic. For example, a user may test two fundamentally different allocation algorithms, such as a greedy heuristic and a genetic algorithm, under identical threat scenarios by simply switching the input configuration, without modifying any simulation logic. Similarly, a planner may develop a custom reporter that tracks cumulative jammer rotation angles to assess long-term mechanical wear, or another that logs algorithm execution time to evaluate real-time feasibility. This plug-and-play architecture decouples algorithm development from simulation engineering, significantly reducing overhead for experimentation and encouraging a more iterative, test-driven approach to algorithm evaluation.

2. Related Work

2.1. Simulation for Test and Evaluation

Simulation is an important tool for testing and evaluating systems, particularly in such fields as defense, where physical testing can be expensive and risky. By developing computer models that replicate various conditions, threats, and scenarios, simulations help identify vulnerabilities and optimize system performance in a safe, controlled, and cost-effective environment [7, 8]. There are three commonly used simulation strategies: activity scanning, process interaction, and event scheduling [9]. However, object-oriented simulation provides a more comprehensive approach by modeling entities (such as algorithms, people, or machines) as objects with unique characteristics and behaviors, facilitating understanding and development of the model [10].

The success of a simulation depends on several key factors, including realism, accuracy, consistency, adaptability, and efficiency [7]. Simulations allow analysts and planners to study the impact of these algorithms under various scenarios [8]. As a result, simulations are applied in such diverse areas as autonomous robots [11], wireless communication [12], cooperative intelligent transport systems [13], and autonomous vehicles [14].

2.2. Threat Evaluation and Weapon Allocation

The Threat Evaluation and Weapon Allocation (TEWA) problem is a critical optimization challenge in defense systems, where the goal is to minimize the expected value of surviving threats by prioritizing and assigning the most suitable countermeasures [15]. TEWA is commonly studied within command and control (C2) systems, where various methods such as neural networks, genetic algorithms, simulated annealing, and taboo search are used to solve the problem [16, 17]. Threat evaluation involves determining the priority of threats based on factors like proximity, capability, and intent, which helps in deciding the appropriate action against each threat [18].

Weapon allocation is about efficiently assigning available defense systems to respond to threats. This problem is classified as NP-hard, meaning that the computational time for any ideal solution increases exponentially with problem size [19, 20]. In practice, meta-heuristic approaches are preferred due to the

complexity of finding exact solutions. In specific applications, such as assigning jammers to angles rather than directly to targets, problem-specific algorithms are required because a jammer can impact multiple targets within a given angle. These specialized algorithms vary based on the characteristics of the anti-drone system and potential types of attacks, necessitating a tailored approach for each scenario [21, 22].

2.3. Agent Based Simulation

Agent-based simulation (ABS) provides a flexible and dynamic method for modeling complex systems by allowing individual agents to act independently and interact with one another, capturing the system's collective behavior and outcomes [23]. Unlike traditional simulations that rely on fixed rules or equations, ABS is well-suited for representing real-world uncertainties and complexities, as agents evolve over time, delay actions, and decide when to take initiative. This approach is particularly advantageous for modeling complex systems because it allows for modular structures, where agents' behavior rules can be easily modified to test different scenarios [24].

ABS has gained popularity in UAV simulations and defense applications because it facilitates coordination and cooperation among units, allowing dynamic and adaptive solutions to complex problems such as TEWA [25, 26]. In TEWA problems, ABS can simulate the tactical decisions of weapons and targets under varying conditions, providing a framework to develop robust defense tactics. Various frameworks such as MASON [6], JADE [27], and Repast Simphony [28] are used for ABS, each offering unique advantages depending on the specific requirements of the simulation study. In this study, MASON was chosen due to its open-source nature, documentation, and level of control it offers over the simulation.

3. JASPER Modeling and Design

JASPER is a specialized simulation platform designed to evaluate jammer allocation algorithms in anti-drone systems. Built on the agent-based simulation framework MASON, JASPER facilitates detailed modeling of interactions between UAVs and jammers, providing a controlled environment for testing and refining these algorithms.

A key feature of JASPER is its modular design, which allows users to integrate different algorithms and reporting agents seamlessly. The platform focuses on executing these integrated components, enabling analysts and planners to observe the behavior of jammer algorithms under various scenarios. The performance metrics, such as coverage percentages and response times, are generated by user-defined reporter agents, which can be customized and incorporated into JASPER. This modularity ensures that the core simulation environment remains flexible and adaptable to different research requirements.

3.1. Assumptions

Modeling a real-world anti-drone system with all its complexity can be an overwhelming task. To make the problem easier to handle and focus on the crucial elements of the jammer allocation methods, the following simplifications and assumptions were made:

- **2D Aerial View Model**: To simplify the environment representation, an aerial view model was used in the simulation design. This method focuses on the interactions between the jammer and the UAV while simplifying the visualization and coverage tasks. The algorithms intended to be tested also work in this kind of modeling. In simplifying, certain land features, such as topography plant cover, and atmospheric conditions are also ignored.
- **Grid System**: Dealing with real-world coordinates introduces a huge overhead for computations. Thus, a continuous grid system was used to keep the locations of the jammers and tracks. This simplification makes it more straightforward to create test scenarios and to use the algorithms by enabling simple computations of angles and distances.
- **Radar Range**: It was assumed that the radar would detect only in a circular, static manner. The assumption assures consistent behavior across every scenario. Typically, geography, meteorological conditions, speed, and frame of the detected object can all affect radar detection ranges. However, these complications are reduced to allow for a more straightforward and predictable implementation by assuming a static and circular range.
- **Track Movement**: Track movements can often be varied and can be affected by a variety of factors, including its mission, environmental conditions, random deviations, and obstacles. These difficulties are avoided by assuming that tracks moved uniformly, following predetermined waypoints in designated simulation time. Because of this assumption, track behavior is less variable, enabling more planned and easily replicated experiments.
- **Jammer Coverage**: Jammers' coverage areas were represented as idealized segments with set angles. This simplification concentrates on the angular optimization component of the algorithms, avoiding the complications of signal transmission and

interference. Usually, jammers do not cover a set angular range; their jamming signals might spread into adjacent locations. Jammers also operate on specific frequencies, which might alter their coverage and efficacy. However, we excluded frequency information from our assumption, further simplifying the model to focus on angular optimization.

These simplifying assumptions were determined to balance realism with computational efficiency. They ensure that the simulation stays focused on the essence of jammer allocation while remaining manageable and extensible.

3.2. Agents

Track. Track agents represent the data from radar, which can be UAV, suspected UAV, and other unimportant tracks. Suspected UAVs are threats that cannot be identified as UAVs with certainty but are classified as being closest to the UAV in terms of probability. Each track agent has several properties:

- Location: The track's current position, as represented by a Double2D object.
- Waypoints: A list of waypoints that the track will follow, each with a specific arrival time.
- Velocity: The speed and direction of the track's movement.
- Type: The type of track (UAV, suspected UAV, or other).
- Distance to Closest Jammer: Used to calculate the track's proximity to the nearest jammer.

Track agents' position changes depending on their velocity and waypoints, and it calculates the distance to the nearest jammer to determine if the track is coming closer or going farther away. Tracks are constructed once for each scenario and traverse on their own.

Jammer. Jammer agents reflect the simulation's jamming devices. Jammer agents have the following attributes:

- Location: The jammer's fixed position.
- Directed At Angle: The current angle from which the jammer is directed.
- Angle Swivel From Previous Step: The angle swivel from the previous step.

Jammer agents do not change the angle on their own. Instead, they alter orientation according to the algorithm agent's assignments.

Algorithm. The algorithm agent is responsible for calculating the optimal angles for the jammers based on the current state of the tracks and jammers. At each step, the algorithm agent processes the simulation state, which includes all tracks' positions and jammer statuses. It then determines the new optimal angles for the jammers to meet its set objectives. These new angles can be assigned through the interface provided by the jammer class.

Users can provide different algorithms to be used without the simulation needing to know their implementation, as long as they implement the interface IAlgorithmAgent defined by JASPER. This modularity enables analysts and researchers to evaluate alternative algorithmic strategies rapidly, without making any changes to the simulation core. Integration is performed at runtime: the user simply supplies a compiled .jar file along with the fully qualified class name of the algorithm. The simulation dynamically loads the class and invokes it through reflection, eliminating the need for recompilation or manual integration steps. This separation of concerns allows developers to focus entirely on their optimization logic, while using JASPER as a black-box evaluation environment.

Reporter. The reporter agent is responsible for calculating and reporting KPIs at each time step of the simulation. At each step, the reporter agent accesses the simulation state to acquire data and calculates its set of KPIs. They may include but are not limited to coverage percentages, angle swivels, and algorithm execution times. The reporter agent then processes these KPIs in accordance with its design.

Like the algorithm agent, users can provide different reporter agents. Thus, different reporter agents can be used to calculate and report different sets of KPIs, giving control over what metrics the user monitors and how the user handles them. Depending on the user's needs, the reporter agent can write the KPIs to a file, display the KPIs on the screen, or even send the KPIs over a network for remote analysis. The simulation framework does not need to know the specifics of each reporter's actions, as long as they implement the interface IReporterAgent given by JASPER.

This modular design allows users to define highly customized reporter agents tailored to the specific needs of their evaluation objectives. For example, a user interested in system efficiency may implement a reporter that captures algorithm execution time and angle swivel variance, whereas another focused on threat coverage might track UAV engagement ratios or coverage stability over time. Since the reporting logic is fully externalized, integrating domain specific KPIs (such as energy consumption, jammer idle time, or sector redundancy) requires no changes to the simulation core. This decoupling ensures that JASPER

remains flexible and reusable across a wide range of operational contexts, encouraging experimentation with novel evaluation strategies without introducing technical overhead.

3.3. Integration of Components

Algorithm Agent. To integrate an algorithm agent, the user provides a file that includes the algorithm. As both MASON and JASPER are in Java, the given file should be a JAR file. Additionally, users give where JASPER can find the algorithm in this file as a path. The user's algorithm agent must implement JASPER's IAlgorithmAgent interface to ensure compatibility with the simulation environment. With this approach, it is easier to switch out and test various algorithms without having to change the JASPER code.

Reporter Agent. To integrate a reporter agent, similar to the algorithm agent, the user provides a file containing the agent itself. For the same reason as the algorithm agent, the file must be a JAR file and the user also specifies the path within this file where JASPER can locate the reporter agent. The user's reporter agent needs to implement the IReporterAgent interface provided by JASPER to be compatible with the simulation environment. To carry out the required computations, the reporter agent receives the simulation state. Different reporter agents can use the state to measure different metrics.

Test Cases. The test cases include information regarding jammers and tracks used in the simulation.

For jammers, the input specifies the number of jammers, along with details such as their positions (X and Y coordinates) and their initial orientation. These parameters define the setup for each jammer within the simulation.

Regarding tracks, the input contains the number of tracks, the total time steps in the simulation, and the type of each track. Additionally, it provides data on the number of waypoints for each track. A waypoint refers to a specific location the track is expected to reach during the simulation. For each waypoint, the input also specifies the corresponding simulation time step at which the track is at that location. The sequence of these waypoints and their associated times is used to determine the movement pattern and timing of the track throughout the simulation.

Users can create several test scenarios to try different kinds of attacks. Thus, it is possible to simulate distinct attack methods and enable the users to see the way many variables interact under different scenarios. This variation in scenario design helps to discover possible weaknesses in algorithms and test them in a controlled, repeatable way, in addition to improving the realism of the simulations.

4. Case Study

In this section, two use cases to demonstrate how the JASPER simulation environment can be used to evaluate a jammer allocation algorithm is presented. For the test cases, an analyst who wants to evaluate how the algorithm presented in [21] behaves in certain scenarios.

4.1. Evaluating Algorithm Behavior in Mixed Confidence Scenarios

For this study, assume an analyst want to examine a scenario in which some UAVs are detected as suspected UAVs. In real-life usage, frequent device rotations in a short time may increase maintenance requirements, or slower assignments relative to track updates might require speeding up the algorithm's decision making time, even if it means sacrificing performance. To analyze these trade offs, the analyst creates a reporter agent that records coverage percentages, angle swivel, and algorithm execution time, using these KPIs to decide if adjustments are necessary. The analyst also creates a test scenario in which, there are two jammers, four UAVs, two suspected UAVs, and eleven unimportant tracks. In the scenario, paths for UAVs and suspected UAVs are hand crafted considering the initial positions, speeds, and movement patterns of the UAVs and suspected UAVs. The representation of initial assignments of jammers in JASPER can be seen in Figure 1.

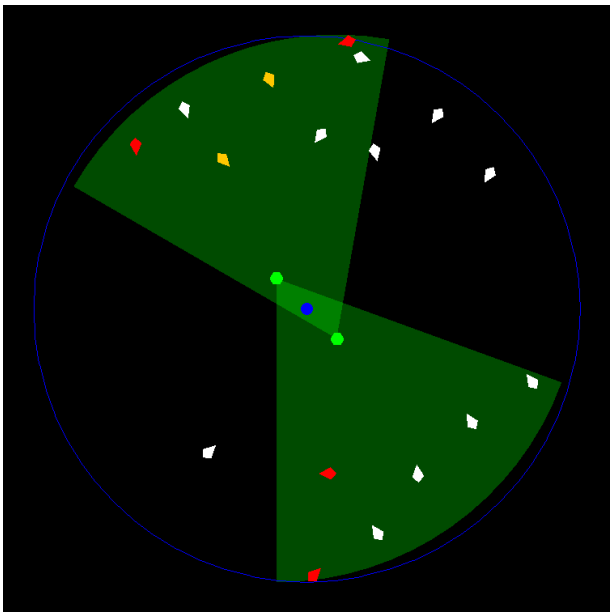

Figure 1. Simulation state after the initial assignment has been made

During the simulation, coverage percentages can be visualized in a chart. The algorithm's performance

was evaluated using the UAVs' coverage percentages, suspected UAVs' coverage percentages, and overall coverage percentages.

The algorithm often produced high UAV coverage percentages throughout the experiment, decreasing only to 75% under circumstances where all UAVs could not be covered. Figure 2 represents a real time visualization generated by JASPER during the simulation process, showcasing coverage percentages over simulation time. It demonstrates the algorithm's performance and variations in coverage percentages during the test scenario.

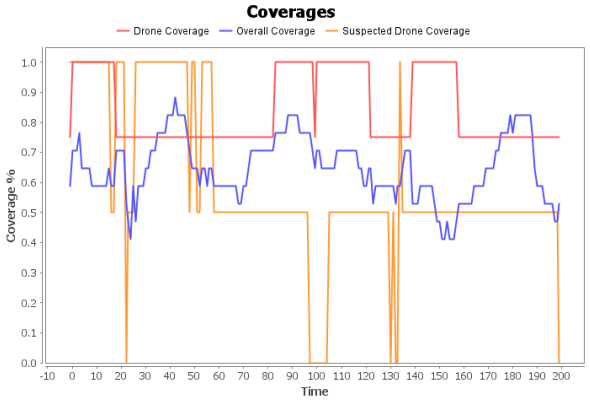

Figure 2. Coverage percentages of the algorithm in the case study (1.0 corresponds to 100%.)

While coverage performance is important, an algorithm needs to take acceptable amount of time to calculate angles. Otherwise, it cannot be used in real-world applications. For that reason, the time taken by the algorithm to compute new angle assignments at each step was recorded. This metric indicates how long the algorithm took to make new angle assignments at each step. In this case, the reporter agent selected includes this metric and the graph of the time taken for the algorithm to calculate can be seen in Figure 3 obtained from the JASPER.

4.2. Evaluating Algorithm Under Jammer Failures

Previous analyses in this paper assumed that all jammers remain operational for the entire duration of a scenario. In practice, however, hardware malfunctions or deliberate electromagnetic counter–measures may render a device inoperative at an arbitrary instant. To evaluate the resilience of the allocation algorithm under such conditions, an additional case study is conducted in which each jammer is subject to a single, random outage time that is generated deterministically from the simulation seed.

The scenario is independent from the previous case study of Section 4.1. The scenario includes four UAV tracks, two suspected UAV tracks and eight non-threat

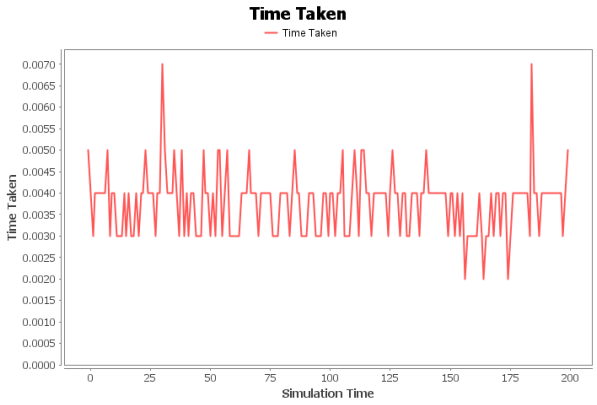

Figure 3. The time taken for the algorithm to calculate the jammer–angle assignments, in seconds

tracks. At simulation start, the pseudo–random number generator draws an outage time in the second quarter of the scenario. Once t_{off} is reached, the affected jammer stops jamming. Two seeds are investigated. For **Run-A** (seed 42) jammer 2 fails at step 64, whereas for **Run-B** (seed 84) jammer 3 fails at step 72. All remaining parameters, including the track set and the algorithm, are identical across the two executions.

The reporter module records the same five KPIs from subsection 4.1. Figure 4 and Figure 5 show the change over time of the three coverage metrics; Figure 6 and Figure 7 show the final simulation state for both seeds.

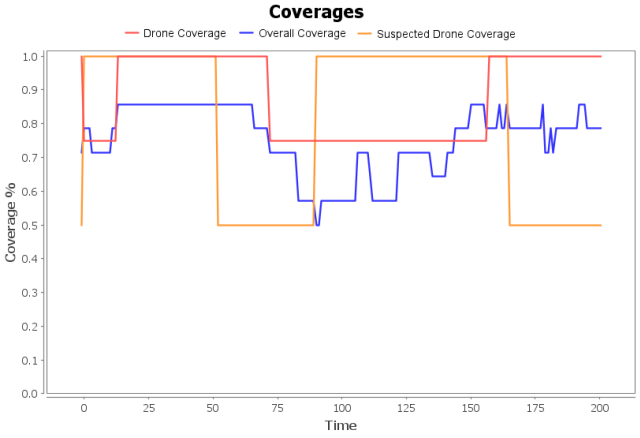

Figure 4. Coverage ratios for Run–A (seed 42)

In Run–A the loss of jammer 2 triggers a sudden drop of the UAV coverage to 0.72 at $t = 66$. The algorithm restores full protection after twenty further steps by reorienting the remaining jammers. During this interval, overall coverage reaches a minimum of 0.50 because several non-threat tracks moves into the newly created blind sector.

Run–B exhibits a similar behavior: the outage at $t = 72$ lowers the UAV coverage to 0.75 and the overall

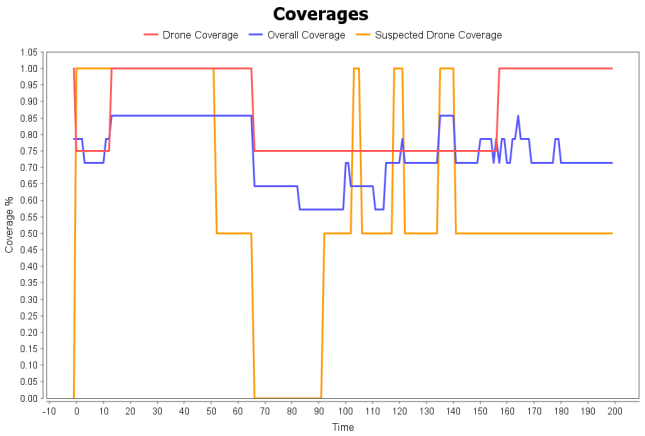

Figure 5. Coverage ratios for Run–B (seed 84)

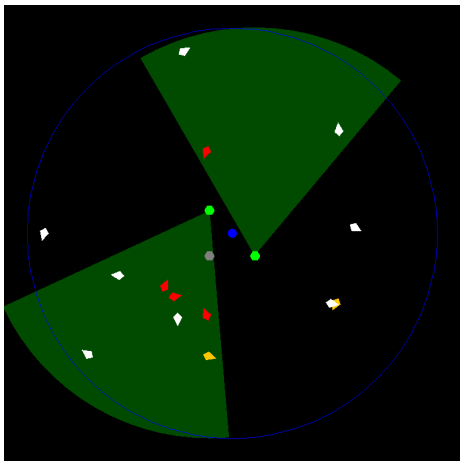

Figure 6. The final simulation state of Run–A

coverage to 0.56, followed by a recovery phase of approximately thirty steps.

Suspected UAV coverage is markedly more volatile in both seeds, oscillating between 0.50 and 1.00 as the algorithm assigns lower priority to ambiguous targets once all UAVs are jammed.

The experiment confirms that a single jammer outage causes a temporary coverage drop, yet the algorithm compensates for this loss after a while. While high–priority UAV tracks regain full protection quickly, suspected UAV tracks remain partially uncovered for extended periods, suggesting that an adaptive weight in the objective function could improve balanced coverage.

5. Evaluations

5.1. Scaling Analysis

The simulation works smoothly with a regular amount of jammers and tracks, with no noticeable performance issues. For testing purposes, the simulation was run with a more crowded real-world scenario, including 150

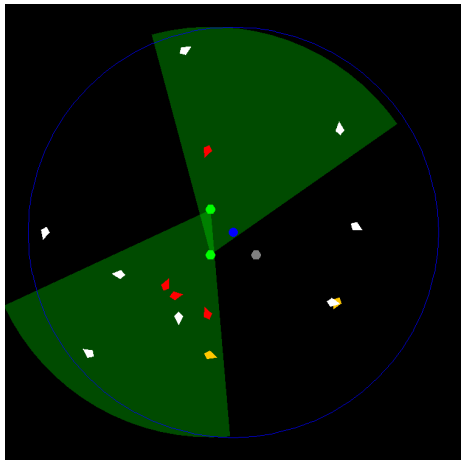

Figure 7. The final simulation state of Run–B

tracks and 10 jammers, and it continued to function normally without interruption.

Given that the simulation handles tracks and jammers equally, they are treated the same during stress testing. During tests, it was found that the simulation can handle 512 entities with little to no lag, while performance may vary based on the settings chosen. However, some MASON features, such as inspectors, can struggle when too many entities are gathered. When the number of entities approaches around 4000, the simulation takes more time to calculate the next state. Despite the additional computing load, it continues to function successfully. The largest number of entities tested was 16384, and while it had some slowdown, the simulation performed successfully. To see the effect of the visualization, the simulation was run on the console under the same conditions. Although there was some speedup, there was not much gain compared to the visualized version. Results of the scaling analysis can be seen in Table 1. Tests were conducted on a computer with an AMD Ryzen 5 7600 processor and 32GB of RAM. For each test, the tested scenario took 100 steps to complete and only execution time is recorded (i.e. no contribution from the algorithm or reporter).

Table 1. Execution time for various entity counts and visualization modes

Entity Count	Execution Time (in seconds)	Execution Time without visualization (in seconds)
2	3.752	3.567
16	3.885	3.632
512	3.923	3.749
1024	4.159	3.893
4096	4.968	4.013
16384	11.106	6.515

It's worth noting that, while the simulation scales well, the algorithm used may have limitations. The algorithm's optimizer might reach limits in terms of the amount of formulations or variables it can handle.

Furthermore, real-time viewing of all trails drastically degrades performance. Even with as few as 30 tracks, displaying all trails can make the simulation incredibly slow and practically unusable. For that reason, the simulation only shows the trail of the selected track, one at a time.

5.2. Sensitivity Analysis

Validation in simulation for testing and evaluation is important to ensure the accuracy and reliability of the model. Without proper validation, the results produced by the simulation may not reflect the true behavior of the system, leading to incorrect conclusions or decisions.

Ideally, validation would involve direct comparison with real-world data. This method provides the highest level of accuracy, as it directly correlates the simulation results with actual observations. However, such comparisons are often expensive and challenging due to the difficulty of obtaining accurate and comprehensive real-world data. Given these challenges, in this study, sensitivity analysis is employed as a practical alternative for validation. This approach examines how variations in input parameters affect the model's output, allowing for an assessment of the simulation's robustness.

Sensitivity Analysis on Algorithm. Firstly, a sensitivity analysis was conducted by running the jammer assignment algorithm at different intervals within the simulation. Using the same test scenario from subsection 4.1, instead of executing the algorithm at every time step, it was tested at intervals of every 2, 3, 4, and 10 time steps. This variation showed the impact of reduced frequency on KPIs.

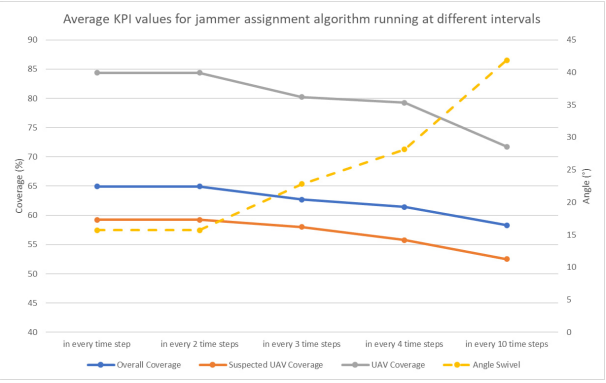

Figure 8. Average KPI values for jammer assignment algorithm running at different intervals

The results of this analysis are illustrated in Figure 8. As expected, the interval between algorithm executions increased, there was a slight decrease in coverage percentages, UAV coverage being the one that fell the most. Additionally, the angle swivel showed a significant increase. In addition to this figure, it is worth noting that the execution time of the algorithm remained constant across all intervals.

This sensitivity analysis revealed that as the algorithm was executed at larger intervals, corresponding declines in performance metrics were observed. This indicates that changes in the frequency of algorithm execution directly impact the simulation outcomes.

Sensitivity Analysis on Jammer Count. This sensitivity analysis investigates how the system's KPIs respond to changes in the number of active jammers, while all other variables remain fixed.

In this experiment, total of six potential jammer sites were available. For tests with fewer devices the configuration was obtained by removing the highest–index jammer(s) so that location of jammers with lower index remained unchanged. All remaining parameters followed the case study of Section 4.1, including the pre-defined flight paths of four UAVs, two suspected UAVs and eleven non-threat tracks. The assignment algorithm was executed once per simulation step, received the full state of the simulation and returned a complete set of jammer–angle assignments. Each configuration was simulated for 200 steps; step-wise values of the five KPIs: overall coverage, suspected-UAV coverage, UAV coverage, execution time of the algorithm, and mean angle-swivel were averaged over the run.

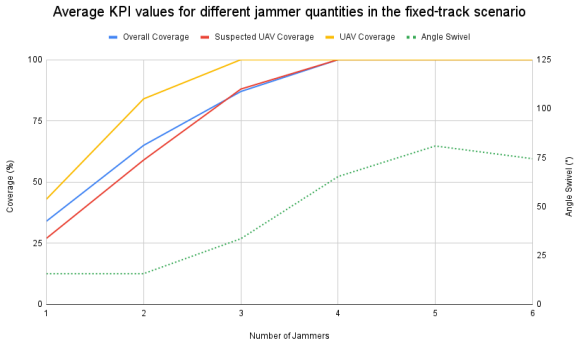

Figure 9. Average KPI values for different jammer quantities in the fixed-track scenario

Figure 9 summarises the outcomes. All three coverage metrics rise almost linearly up to $n = 4$. At that point UAV coverage reaches 100%, while overall coverage attains a comparable value when a fifth jammer is introduced. Beyond four devices the curves plateau,

indicating that additional jammers yield only marginal gains.

In contrast, the mean angle-swivel increases almost monotonically up to $\approx 80°$ for $n = 5$. This increase is due to the algorithm's tendency to take advantage of each new available jammer. As the number of jammers increases, multiple jammers are trying to jam the same tracks simultaneously across the tracks' movements, resulting in a higher average swivel per device. When a sixth jammer is activated, the combined field of view already covers almost the entire field; further rotations provide little incremental benefit. The resulting saturation—together with the greater sector overlap—reduces the need for individual adjustments and brings the average swivel down to $\approx 75°$ for $n = 6$ likely due to increased overlap between areas jammers cover, which reduces the individual adjustment demands. Algorithm execution time per step remained approximately constant across different jammer quantities.

For the present geometry, four jammers ensure full protection of high-priority UAV tracks and leave only a minor uncovered fraction of lower-priority tracks. A fifth jammer closes this gap but increases mechanical workload by roughly 25 %. A sixth unit adds no further coverage benefit. The smooth progression of coverage and angle swivel (together with the stable execution times) confirms that both the simulation framework and the integrated allocation algorithm respond predictably to variations in resource levels, reinforcing confidence in JASPER's suitability for test-and-evaluation studies.

6. Conclusion

In this study, JASPER, a modular, agent-based simulation testbed was developed to evaluate jammer-angle allocation algorithms for anti-drone systems. JASPER simulation environment, built using the MASON framework, supports the integration and comparison of different algorithms across various scenarios. Key features include a configurable reporter agent and the ability to create diverse test cases. For observing how variations in input parameters affect the model's output, a sensitivity analysis is performed to JASPER. Moreover, for observing the performance under different numbers of jammers and tracks, a scaling analysis is also performed. Simplifications like ideal jammer coverage and a 2D aerial view were made to focus on essential elements, though these also introduced some limitations. These simplifications were deliberately chosen to minimize their impact on its application to real-world scenarios. If the real-world modeling of the algorithm to be tested, significantly differs from these assumptions, this tool may not be suitable for testing such algorithms.

Future improvements to the simulation environment could address several limitations and enhance its realism. For instance, evaluating the effects of multiple jammers' interference and supporting different jammer coverage patterns could provide a more accurate assessment of jammer effectiveness. Additionally, future research could incorporate frequency-specific jamming capabilities, reflecting the diverse operating frequencies of different UAV types. Moreover, developing a tool for automatic test case generation could make the simulation more user-friendly. Such improvements would give analysts and planners in the field of anti-drone systems a more complete tool for the evaluation of jammer allocation methods.

References

[1] Shi, X., Yang, C., Xie, W., Liang, C., Shi, Z. and Chen, J. (2018) Anti-drone system with multiple surveillance technologies: Architecture, implementation, and challenges. *IEEE Communications Magazine* **56**(4): 68–74.

[2] Park, S., Kim, H.T., Lee, S., Joo, H. and Kim, H. (2021) Survey on anti-drone systems: Components, designs, and challenges. *IEEE access* **9**: 42635–42659.

[3] Tyurin, V., Martyniuk, O., Mirnenko, V., Open'ko, P. and Korenivska, I. (2019) General approach to counter unmanned aerial vehicles. In *2019 IEEE 5th International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD)* (IEEE): 75–78.

[4] Carling, R.L. (1993) A knowledge-base system for the threat evaluation and weapon assignment process. *Naval Engineers Journal* **105**(1): 31–41.

[5] Karasakal, O. (2008) Air defense missile-target allocation models for a naval task group. *Computers & Operations Research* **35**(6): 1759–1770.

[6] Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K. and Balan, G. (2005) Mason: A multiagent simulation environment. *Simulation* **81**(7): 517–527.

[7] Balci, O. (1995) Principles and techniques of simulation validation, verification, and testing. In *Proceedings of the 27th conference on Winter simulation*: 147–154.

[8] Choi, S.Y., Park, K., Yang, J.H. and Kang, H.I. (2015) Application of agent-based multimethod simulation approach to the simulation testbed prototype for the concept exploration and requirement analysis of ugv. In *ECMS*: 57–63.

[9] Carson, J.S. (2005) Introduction to modeling and simulation. In *Proceedings of the Winter Simulation Conference, 2005.* (IEEE): 8–pp.

[10] Rothenberg, J. (1986) Object-oriented simulation: Where do we go from here? In *Proceedings of the 18th conference on Winter simulation*: 464–469.

[11] Puzicha, A. and Buchholz, P. (2021) A simulation environment for autonomous robot swarms with limited communication skills. In *Simulation Tools and Techniques: 12th EAI International Conference, SIMUtools 2020, Guiyang, China, August 28-29, 2020, Proceedings, Part II 12* (Springer): 206–226.

[12] Cai, W., Song, X., Liu, C., Jiang, D. and Huo, L. (2021) An adaptive and efficient network traffic measurement

method based on sdn in iot. In *International Conference on Simulation Tools and Techniques* (Springer): 64–74.

[13] Aramrattana, M., Larsson, T., Jansson, J. and Nåbo, A. (2019) A simulation framework for cooperative intelligent transport systems testing and evaluation. *Transportation research part F: traffic psychology and behaviour* **61**: 268–280.

[14] Alghodhaifi, H. and Lakshmanan, S. (2021) Autonomous vehicle evaluation: A comprehensive survey on modeling and simulation approaches. *IEEE Access* **9**: 151531–151566.

[15] Hosein, P.A. (1989) *A class of dynamic nonlinear resource allocation problems*. Ph.D. thesis, Massachusetts institute of technology.

[16] Athans, M. (1987) Command and control (c2) theory: A challenge to control science. *IEEE Transactions on automatic control* **32**(4): 286–293.

[17] Cai, H., Liu, J., Chen, Y. and Wang, H. (2006) Survey of the research on dynamic weapon-target assignment problem. *Journal of Systems Engineering and Electronics* **17**(3): 559–565.

[18] Johansson, F. and Falkman, G. (2008) A bayesian network approach to threat evaluation with application to an air defense scenario. In *2008 11th International conference on information fusion* (IEEE): 1–7.

[19] Lloyd, S.P. and Witsenhausen, H.S. (1986) Weapons allocation is np-complete. In *1986 summer computer simulation conference*: 1054–1058.

[20] Hosein, P.A., Athans, M. *et al.* (1990) Some analytical results for the dynamic weapon-target allocation problem .

[21] Barakli, A.B., Semiz, F. and Atasoy, E. (2023) The specialized threat evaluation and weapon target assignment problem: Genetic algorithm optimization and ilp model solution. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)* (Springer): 19–34.

[22] Gül, K. (2018) *Model and procedures for the jammer and target allocation problem*. Master's thesis, Middle East Technical University.

[23] Borshchev, A. and Filippov, A. (2004) From system dynamics and discrete event to practical agent based modeling: reasons, techniques, tools. In *Proceedings of the 22nd international conference of the system dynamics society* (Oxford, England), **22**: 25–29.

[24] Macal, C. and North, M. (2014) Introductory tutorial: Agent-based modeling and simulation. In *Proceedings of the winter simulation conference 2014* (IEEE): 6–20.

[25] Mualla, Y., Bai, W., Galland, S. and Nicolle, C. (2018) Comparison of agent-based simulation frameworks for unmanned aerial transportation applications. *Procedia computer science* **130**: 791–796.

[26] Lorig, F., Dammenhayn, N., Müller, D.J. and Timm, I.J. (2015) Measuring and comparing scalability of agent-based simulation frameworks. In *Multiagent System Technologies: 13th German Conference, MATES 2015, Cottbus, Germany, September 28-30, 2015, Revised Selected Papers 13* (Springer): 42–60.

[27] Bellifemine, F., Poggi, A. and Rimassa, G. (1999) Jade–a fipa-compliant agent framework. In *Proceedings of PAAM* (London), **99**: 33.

[28] North, M.J., Collier, N.T., Ozik, J., Tatara, E.R., Macal, C.M., Bragen, M. and Sydelko, P. (2013) Complex adaptive systems modeling with repast simphony. *Complex adaptive systems modeling* **1**: 1–26.