Robust Communication-Aware Jamming Detection and Avoidance for Distributed Multi-Agent Systems[⋆]

Sang Xing[a,*], Samuel Peccoud[a,1,*], Thomas Yang[a], Richard Stansbury[a]

[a]*Embry-Riddle Aeronautical University, 1 Aerospace Boulevard, 32114, Daytona Beach, Florida, USA*

Abstract

Swarm robotics has emerged as an innovative field, enabling groups of agents to collectively tackle complex challenges. In this pursuit, Multi-Agent Systems (MAS) offer unparalleled potential for navigating through intricate environments. In this paper, an innovative control strategy is proposed to enable a swarm of agents to navigate through a complex environment, which includes regions affected by electromagnetic jamming. The swarm is unaware of the existence or location of these regions, known as jamming areas. The novel control strategy is proposed as a combination of a gradient and movement controller. The gradient controller achieves the desired swarm formation that maximizes the communication quality between agents. The movement controller uses Particle Swarm Optimization (PSO) and a path planning algorithm to move towards a destination while avoiding jamming areas. Various path planning algorithms, such as A*, Greedy Best First Search, Breadth First Search, Dijkstra, Jump Point Search, and Theta*, are compared in simulations. The results highlight the discovery of an optimal path planning algorithm that facilitates efficient and robust navigation for the MAS, while also upholding high communication quality.

[⋆]This research was supported by the National Science Foundation under Grant No. 2150213.

[*]These authors contributed equally to this work.

Email addresses: `xings@my.erau.edu` (Sang Xing), `speccoud@colostate.edu` (Samuel Peccoud), `yang482@erau.edu` (Thomas Yang), `stansbur@erau.edu` (Richard Stansbury)

[1]Author has moved to Colorado State University since the work described in the article was done.

Keywords: multi-agent systems, communication-aware, formation control, particle swarm optimization, path planning, jamming area

1. Introduction

The advent of distributed MAS has sparked a paradigm shift in various fields, revolutionizing autonomous robotics, environmental monitoring, disaster response, and communication networks. These decentralized networks, composed of interconnected agents, exhibit collective intelligence and efficiency, empowering them to collaboratively address intricate tasks and navigate dynamic environments [1]. However, as the reliance on distributed MAS grows, it also unveils new challenges, particularly in the realm of modern electronic warfare.

Electronic warfare encompasses a diverse range of tactics, and among them, jamming emerges as a formidable threat that jeopardizes the seamless functioning of distributed MAS [2]. Jamming activities focus on disrupting communication channels, causing disruptions in the exchange of vital information among agents and leading to coordination failures and mission setbacks. In this context, developing robust jamming detection and avoidance strategies becomes paramount to ensure the dependability and effectiveness of these systems in hostile environments.

This research paper proposes a novel control strategy for MAS using a communication-aware gradient controller, Particle Swarm Optimization (PSO), and path planning algorithms. In addition to demonstrating the robustness of this control strategy, six different path planning algorithms are compared and evaluated to determine which one is best for this application.

The paper is organized into distinct sections, each addressing different aspects of the proposed control strategy. To begin, Section 2 offers an extensive overview of the preparatory work necessary for comprehending the suggested control approach. Subsequently, Section 3 introduces the interaction model established at the communication layer. In Section 4, the construction of the control layer is detailed, incorporating PSO and multiple path planning algorithms. The seamless integration of the communication layer and control layer into a new control strategy is described in Section 5. Moving forward, Section 6 presents the simulation environments and process. An evaluation of these results follows in Section 7. Lastly, Section 8 concludes the paper.

2. Preliminaries

2.1. System Model

Consider a collective of n single-integrator modeled agents navigating within a two-dimensional space. The behavior and movement of each agent in time are governed by their respective dynamics, which can be described as follows:

$$\dot{q}_i(t+1) = q_i(t) + z_i, \quad i = 1, 2, \dots, n, \tag{1}$$

where $q_i, z_i \in \mathbb{R}^2$, $q, z \in \mathbb{R}^{2n}$, and $i \in \mathcal{V}, \mathcal{V} = \{1, 2, \dots, n\}$,

- q_i denotes the position of i-th agent at time t.
- z_i denotes the control input of i-th agent.

2.2. Graph Theory

Graph theory has emerged as a widely adopted approach for modeling MAS [3]. A graph G can be represented as a pair $(\mathcal{V}, \mathcal{E})$, comprising a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$. The vertices are typically denoted by agent $1, 2, \dots, n$, and the edges consist of ordered pairs of vertices, denoted as $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The classification of graphs is based on various properties they exhibit.

To facilitate the analysis of a specific agent i, the neighboring set of i, denoted as N_i, is defined as $N_i = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}\}$, encompassing all vertices that are directly connected to i via an edge in the graph.

2.3. Rigid Formation

The decentralized control of rigid body formations has been a subject of extensive investigation among researchers, often incorporating concepts from graph rigidity theory and artificial potential fields [4]. In a rigid formation, the inter-agent distances remain constant during motion. Agents perceive the relative positions of their neighbors using local coordinate systems. Specifically, the relative position vector between agent i and agent j is denoted as $\vec{q}_{ij} = q_i - q_j$ [5], and the relative distance is mathematically expressed as

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = \|q_i - q_j\|, \tag{2}$$

where x_i, y_i represent the coordinates of agent i [5]. To address communication complexity and mitigate potential errors, a communication range R is introduced. Within a two-dimensional open ball with a radius of R, the neighboring set of agent i is defined as $N_i = \{j \in \mathcal{V} \mid r_{ij} \leq R\}$, encompassing all agents located within a distance of R from agent i.

3. Communication Layer

3.1. Communication-aware Interaction Model

In the communication layer of a distributed system, the quality of transmissions between agents is crucial. To assess the channel quality, the communication-aware interaction model, denoted as $\phi(r_{ij})$, is the product of far-field reception probability a_{ij} and the near-field propagation factor g_{ij} [6].

3.2. Gradient Controller

To achieve distributed rigid formation control, a gradient-based control law is employed. The objective is to manage the inter-agent distances between agents in a way that optimizes the quality of communication. To do this, an artificial potential function $\psi(r_{ij})$ is introduced to model the interaction between agent i and agent j [7].

In [6], the author defines the gradient-based controller $\mathcal{G}_i$ as the negative gradient of the local potential functions $\psi_t(r_{ij})$:

$$\mathcal{G}_i = -\nabla_{q_i} \left[\sum_{j \in N_i} \psi_t(r_{ij}) \right], \tag{3}$$

By imposing a gradient controller on the communication-aware interaction model, the MAS can achieve efficient and reliable communication between agents. The gradient controller influences the movement of the agents creating the swarm formation.

4. Control Layer

4.1. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a powerful computational optimization technique inspired by the collective behavior of bird flocking and fish schooling [8]. It is used in combination with the gradient controller to provide movement to the swarm. In PSO, a population of particles, each representing a potential solution, moves through the problem space while adapting their velocities based on their own best-known position and the best position discovered by any particle in the swarm. This cooperative exploration and exploitation process allows PSO to rapidly converge towards promising regions of the search space, making it particularly useful for solving complex, high-dimensional, and non-linear optimization problems. Its simplicity, effectiveness, and ability to handle dynamic environments have made PSO a popular choice in many practical applications [9].

An adaptation of PSO is proposed allowing agents to avoid jamming areas. First, each agent in the swarm is modeled as a particle [10]. Then, a fitness function is used to determine which agent is in an optimal location within the environment. This serves as a ranking system, identifying which agent is closer to the optimal solution. The fitness function is defined as

$$f_i = d_{\text{dest}} \cdot w_{\text{dest}} - \log_{10}(h_{\text{jam}}) \cdot w_{\text{jam}}, \tag{4}$$

$$d_{\text{dest}} = \sqrt{(x_i - x_{\text{dest}})^2 + (y_i - y_{\text{dest}})^2}, \tag{5}$$

$$d_{\text{jam}} = \sqrt{(x_i - x_{\text{jam}})^2 + (y_i - y_{\text{jam}})^2}, \tag{6}$$

$$h_{\text{jam}} = \begin{cases} d_{\text{jam}}, & d_{\text{jam}} < 70 \\ 70, & \text{otherwise} \end{cases}. \tag{7}$$

f_i considers the distance to the destination point, d_{dest}, and the distance to the nearest jam point, d_{jam}. The agent with the lowest fitness value is considered to be in the best location. An adjustment made to the fitness function is the logarithmic scaling of the distance to the closest jam point. This adjustment emphasizes jam points that are close and ignores them when far away. Additionally, the system imposes a constraint on d_{jam}, limiting it to a maximum of 70 units, as shown in h_{jam}. w_{dest} and w_{jam} are constant values that can be tuned to weigh the distance to the destination or the closest obstacle [11].

The fitness of i-th agent is used to maintain two important values during the optimization process: the global best value, G_{best}, and the personal best value, P_i^{best}. The global best value represents the best fitness value among any agent in the swarm. It serves as the benchmark for all agents, guiding them towards promising regions in the environment. On the other hand, the personal best value is the best fitness value that an individual agent has found throughout the simulation. This value enables that agent to explore and exploit locally optimal regions [12]. G_{best}, P_i^{best}, and the agents' previous velocity, V_i^{prev}, are used to direct the movement of agents in the swarm. The previous velocity is denoted as

$$V_i^{\text{prev}} = q_i(t) - q_i(t-1). \tag{8}$$

The previous velocity is calculated by subtracting the agent's current position, $q_i(t)$, from the agent's previous position, $q_i(t-1)$.

The velocity for the i-th agent generated by the PSO algorithm is defined as,

$$\begin{aligned} \mathcal{P}_i = & V_i^{\text{prev}} \cdot w_{\text{vel}} \\ & + P_i^{\text{best}} \cdot c_{\text{personal}} \\ & + G_{\text{best}} \cdot c_{\text{global}}. \end{aligned} \tag{9}$$

$\mathcal{P}_i$ is tunable using the following constant weight parameters: w_{vel}, c_{personal}, and c_{global} [13].

The PSO controller and the gradient controller are combined to update the position of each agent in the swarm as follows:

$$z_i = \mathcal{G}_i + \mathcal{P}_i. \tag{10}$$

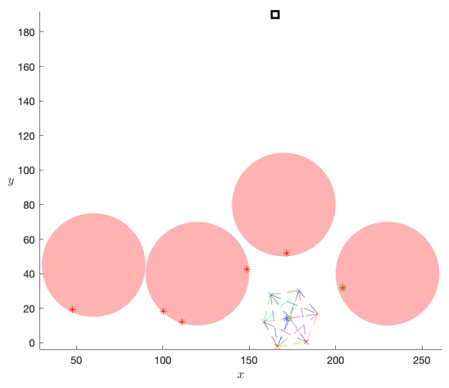
When combined, these two techniques provide motion to a destination and away from jam points all while maximizing communication quality within the swarm. Many simulations were conducted to find optimal control parameters shown in Table 1. The disparity between the values of w_{dest} and w_{jam} arises

Table 1: Control Parameters for Figure 1

w_{dest}	w_{jam}	w_{vel}	c_{personal}	c_{global}
1	110	0.11	0.07	0.13

from the logarithmic scaling of h_{jam} in (7). There is also a difference in c_{personal} and c_{global} because the agents in the swarm are close in proximity. This makes the personal best value less important than the global best value since agents are not exploring regions individually, but rather as a tight-knit swarm.

Many simple environments with only a couple of jamming areas were successfully navigated with this control strategy, but there are issues when the environment increases in complexity. The problem occurs when the PSO controller encounters a deadlock caused by agents being trapped in a local minimum, hindering their ability to reach the global minimum [14]. Figure 1 demonstrates a case where PSO fails to reach the destination denoted as the black square. The swarm is depicted as nodes with dashed lines indicating communication links between agents. The large red circles in the simulation


(a) In deadlock

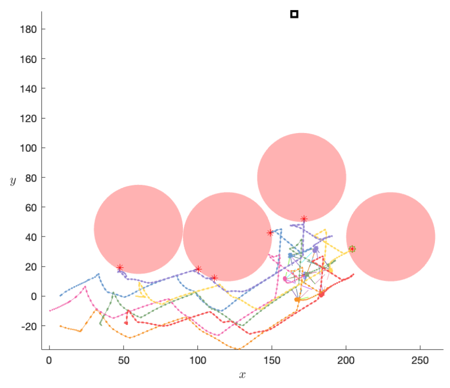

(b) Trajectory of deadlock

Figure 1: Example of deadlock with PSO.

represent unknown jamming areas, and the small red stars represent known jam points, where an agent previously entered a jamming area. Figure 1b shows the trajectory of the swarm further demonstrating the deadlock. Overall, PSO combined with the gradient controller may avoid simple jamming scenarios it is not nearly effective enough to be considered robust.

4.2. Path Planning

To address potential deadlocks that may occur when using only the gradient controller and PSO, path planning algorithms are introduced into the formation control strategy. By incorporating these path planning algorithms, the swarm can effectively navigate around jamming areas, avoiding deadlocks and achieving efficient navigation towards the destination [15].

4.2.1. Grid Map and Grid Cost

To facilitate the path planning algorithms' execution, the environment is discretized into a grid map. The grid map represents the area surrounding the swarm and includes jam points, checkpoints, and the destination. Each grid cell is assigned a cost based on its distance to a jam point.

$$\text{grid cost} = \begin{cases} 10, & \text{if grid distance to jam point} \in (0, 15] \\ 5, & \text{if grid distance to jam point} \in (15, 25] \\ 1, & \text{if grid distance to jam point} \in (25, \infty). \end{cases} \tag{11}$$

Grid cells that are within or close to the jam points are assigned a higher cost, reflecting the increased difficulty and risk associated with navigating in these regions. Conversely, cells that are farther away from the jam points are assigned lower costs, indicating safer and more accessible areas for the swarm to navigate through. The use of grid maps and different grid costs enables the swarm to use path planning algorithms in the discretized environment.

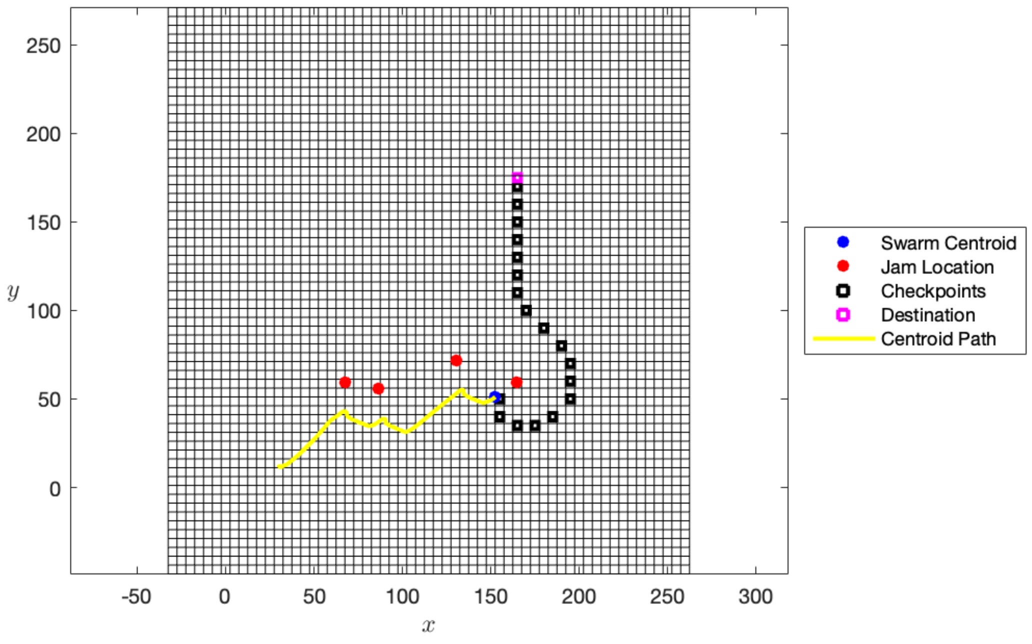


Figure 2: Grid map of path planning algorithm using A*.

As shown in Figure 2, the swarm's centroid is used as the reference point for path planning, and the path planning algorithms produce a series of checkpoints leading to the destination. The agents then follow the checkpoints which guide them to avoid jamming areas while maintaining formation. The planned path is represented by a series of checkpoints that serve as a road map for the agents, providing them with a clear path to follow while avoiding potential deadlocks. Overall, path planning creates a series of checkpoints that enhance agents' navigation and decision-making capabilities.

4.2.2. Integrated Algorithms

The following path planning algorithms are compared and evaluated with the current control strategy (10).

In the context of grid search algorithms, Jump Point Search (JPS) and Theta* are optimization techniques built upon the A* search algorithm, specifically tailored for uniform-cost grids. Grid search problems involve exploring a two-dimensional grid, and these techniques aim to improve efficiency and path quality.

Jump Point Search (JPS) tackles the issue of symmetries in the search process by employing graph pruning. It intelligently eliminates certain nodes in the grid by making assumptions about the neighbors of the current node, following specific conditions based on the grid's geometry. This pruning allows JPS to reduce redundant exploration and expedite the search process [16].

Theta*, on the other hand, enhances the A* algorithm by introducing line-of-sight checks between nodes. This step determines whether some nodes can be skipped during the search, thereby reducing the number of node expansions and ultimately improving the overall path quality [17].

Switching to uninformed search algorithms, Breadth-First Search (BFS) offers a systematic approach to exploring nodes. It traverses the grid in a breadth-first manner, meaning it thoroughly explores all nodes at the current depth level before moving on to the next level. This property guarantees that BFS will find the shortest path to the solution, making it a complete and optimal approach for finding shallow goals in a tree or graph. However, BFS can become computationally expensive when dealing with large search spaces due to the need to store and manage numerous nodes in memory at each level [18].

In contrast to uninformed searches, Greedy Best First Search (GBFS) is an informed algorithm that relies on a heuristic function to estimate the remaining cost to the goal from each node. This heuristic-based approach prioritizes nodes with lower estimated costs, assuming they are more likely to lead towards the desired solution [19]. As a result, GBFS efficiently explores promising areas of the search space, making it particularly valuable when a well-designed and accurate heuristic is available.

Next, we have Dijkstra's algorithm, another uninformed search algorithm. It specializes in finding the shortest path from the starting point to the destination. It does so by exploring nodes based on their total cost from the starting point, gradually building the shortest path tree [20].

Lastly, A* combines the advantages of Dijkstra's algorithm and GBFS. Like GBFS, it employs a heuristic function to estimate the remaining cost, but it also systematically explores nodes to find the shortest path from the starting point to the destination. This combination makes A* a highly efficient and effective informed search algorithm for grid-based problems [21].

5. Novel Control Strategy

To meet the requirements of a robust jamming area detection and avoidance strategy that is communication-aware, the three previously mentioned control methods, gradient controller, PSO, and path planning, are employed together. Figure 3 shows a hierarchical view of the total system. The schematic is split horizontally into two parts, the top half, above the drones and the bottom half, below the drones.

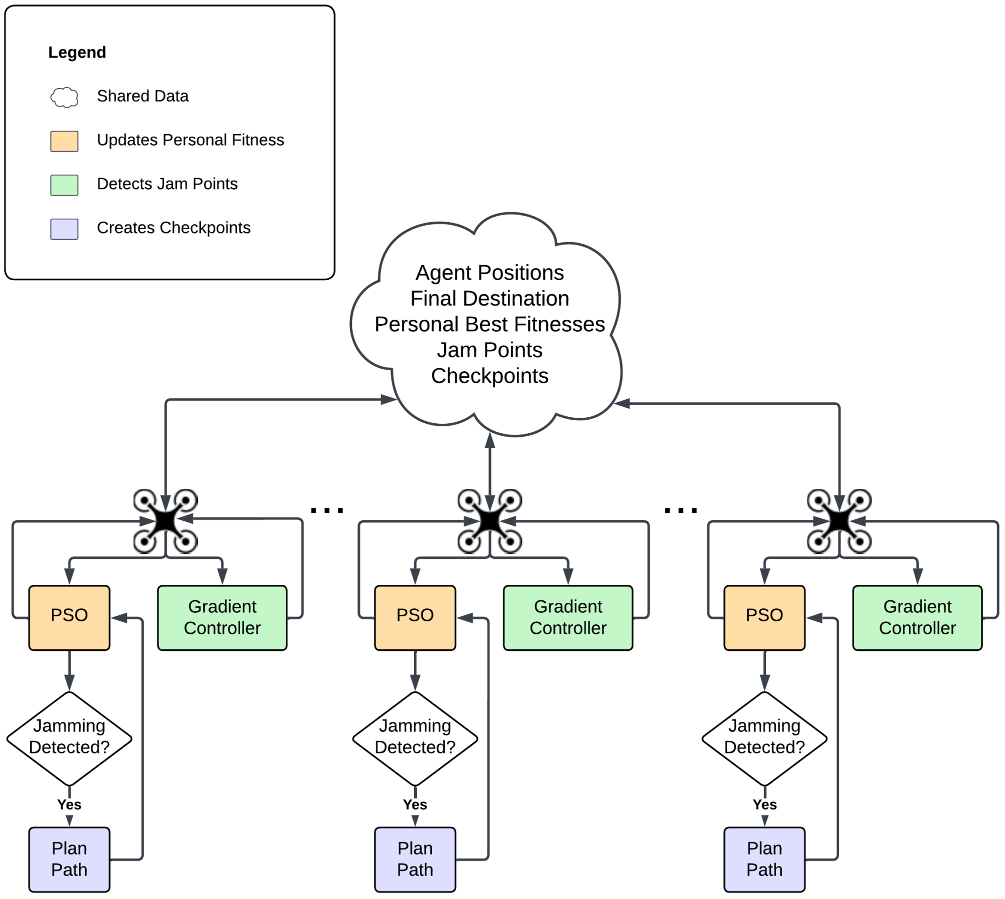


Figure 3: Schematic Diagram of Proposed Control Strategy

The first part, above the drones, illustrates how data is distributed in the system. All the information in the cloud represents the shared data that all the agents have in common. This information is assumed to be up to date so long as one communication link between a drone and the rest of the swarm is active. Each arrow represents data transmission to or from an agent. This data sharing methodology guarantees a distributed approach as it functions independently without relying on any external information to operate the swarm.

In the second part, below the drones, the control flow for each agent is illustrated. Since the MAS is distributed, the control flow is the exact same for each agent and no processing is done on the shared data outside of the local processing by each agent. Coming down from each drone, an arrow splits left and right. The left side represents the Control Layer from Section 3, and the right side represents the Communication Layer from Section 4. These two processes occur in parallel on each agent. The Communication Layer includes the gradient controller as described in Section 3.2 which manages the formation control of the swarm, updating the agent's position based on its communication links. The position updates converge around a formation that maximizes communication quality. As indicated by the legend, the communication layer also detects when an agent enters a jamming area and registers the location of the agent as a jam point. This is then communicated with the rest of the swarm. On the left side, the Control Layer includes the PSO from Section 4.1 and path planning described in Section 4.2. The PSO forms a closed loop with the agent, meaning that the PSO does not require path planning to operate. A path planning algorithm is only executed when an agent in the swarm has detected the presence of a jamming area, shown by the "Jamming Detected?" conditional. The path planning creates checkpoints that the swarm should fly through to reach the destination. These checkpoints are used in the fitness function of the PSO as follows:

$$\begin{aligned} f_i = \sqrt{(x_i - x_{\text{checkpoint}})^2 + (y_i - y_{\text{checkpoint}})^2} \cdot w_{\text{dest}} \\ - \log_{10}(h_{\text{jam}}) \cdot w_{\text{jam}}. \end{aligned} \tag{12}$$

It is similar to the previous fitness function (4), but the destination coordinates, x_{dest} and y_{dest}, are replaced with the coordinates of the next checkpoint, $x_{\text{checkpoint}}$ and $y_{\text{checkpoint}}$. A checkpoint is deemed "checked" when the swarm centroid is approximately within a 20-unit distance from the checkpoint, which roughly corresponds to the radius of the swarm formation. When no path planning has been executed or when path planning fails to identify a feasible path, the fitness function is reverted back to (4).

In a distributed manner, the MAS establishes a formation via the gradient controller while simultaneously progressing towards the next checkpoint along the planned path because of PSO. To accomplish these calculations, data exchange between the agents is essential.

6. Simulation

To evaluate the effectiveness of the proposed formation control strategy with the integrated path planning algorithms, extensive simulations were conducted using MATLAB.

6.1. Simulation Environment Setup

As shown in Figure 4, various simulation environments were created to test the swarm's navigation capabilities and robustness. There are five different environments of increasing complexity, each containing a different number of jamming areas from one to five. Each environment was designed to have one optimal path to the destination and a few sub-optimal paths. The sub-optimal paths were introduced so that there was no obvious solution for the path planning algorithms. The swarm of agents will need to navigate through these environments to reach the destination while avoiding the jamming areas effectively.

6.2. Swarm Navigation Through Jamming Areas

Figure 5 illustrates a simulation run on the environment with five jamming areas, using the A* path planning algorithm. Every simulation conducted follows a similar procedure. Roughly the first ten seconds are used to allow the gradient controller to reach a steady state, like in Figure 5a. Then, the PSO is activated moving the swarm towards the destination unaware of the jamming areas ahead. Eventually, when an agent enters a jamming area a jam point is registered, and the path planning creates checkpoints, represented by black squares in the simulation, as seen in Figure 5b. When the swarm passes the checkpoints they turn green, and the final destination is represented by a magenta square. Each time a new jam point is discovered, the path is recalculated, overwriting the old checkpoints. As the swarm of agents moves through the environment, their trajectory is recorded and shown in Figure 5h. In this example, the swarm was able to reach its destination successfully after encountering only eight jam points, demonstrating the effectiveness of the proposed control strategy.

All the path planning algorithms were tested on every environment, resulting in thirty total simulations.

7. Evaluation of Simulation Results

This section presents the evaluation of the simulation results for the novel control strategy (10) using the six different path planning algorithms: Greedy Best First Search (GBFS), A*, Breadth First Search (BFS), Dijkstra, Jump Point Search (JPS), and Theta*.

7.1. Normalization

Below are the metrics that compare the performance of the path planning algorithms.

1. **Number of Iterations**: The total number of iterations required for the swarm to reach the destination.
2. **Execution Time**: The total execution time required for the swarm to reach the destination.
3. **Average Distance Traveled**: The distance that each agent traveled throughout the simulation averaged.
4. **Average Communication Quality of the Swarm**: The average communication quality of the MAS during the execution.
5. **Number of Jam Points**: The total occurrences of agents entering jamming areas or the number of registered jam points during the simulation.

The metrics are then normalized for each algorithm to ensure a fair comparison across different scales. The normalization formula used for each metric is as follows:

$$\text{Normalized Value} = \frac{x - \min(x)}{\max(x) - \min(x)} \in [0, 1] \tag{13}$$

where x represents the original value of the metric, $\min(x)$ and $\max(x)$ are the minimum and maximum values of the metric among all algorithms in each environment.

7.2. Comparison of Algorithms

After normalizing the metrics, the average is taken for each algorithm in all 5 environments to compare the performance. Figure 6 shows the comparison results in the form of radar charts.

Based on the charts, A* demonstrates the best overall performance among the six path planning algorithms. It outperforms the other algorithms in all six metrics. Therefore, A* is the recommended algorithm for the current model in the given simulation scenarios.

7.3. Overall Performance

The evaluation results offer valuable insights into the individual strengths and weaknesses of each algorithm, facilitating the choice of the most appropriate one. In all thirty simulations, the swarm achieved successful navigation, reaching the destination with a high communication quality index. This confirms the control strategy's robustness and effectiveness, as even less optimal strategies demonstrated satisfactory performance.

8. Conclusion

In conclusion, this paper proposes a novel control strategy for a swarm of agents to navigate through jamming areas efficiently while maintaining formation and robust communication. The strategy combines gradient control, PSO, and path planning algorithms to achieve successful navigation in complex environments. Through extensive simulations and evaluation of six path planning algorithms, A* is identified as the most effective approach. The proposed strategy demonstrates its effectiveness in guiding the swarm through challenging scenarios with multiple jamming areas, ensuring high communication quality. Future work could involve modeling communication procedures and delays to update shared data within the swarm, providing a more realistic representation of swarm behavior, and enhancing the control strategy's capabilities for real-world applications.

References

[1] Kamdar, R.; Paliwal, P.; Kumar, Y. A state of art review on various aspects of multi-agent system. *Journal of Circuits, Systems and Computers* **2018**, *27*, 1830006.

[2] Vadlamani, S.; Eksioglu, B.; Medal, H.; Nandi, A. Jamming attacks on wireless networks: A taxonomic survey. *International Journal of Production Economics* **2016**, *172*, 76–94.

[3] Zhao, Y.; Hao, Y.; Wang, Q.; Wang, Q. A rigid formation control approach for multi-agent systems with curvature constraints. *IEEE Transactions on Circuits and Systems II: Express Briefs* **2021**, *68*, 3431–3435.

[4] Durniak, A. Welcome to IEEE Xplore. *IEEE Power Engineering Review* **2000**, *20*, 12.

[5] Olfati-Saber, R. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control* **2006**, *51*, 401–420.

[6] Li, H.; Peng, J.; Liu, W.; Gao, K.; Huang, Z. A novel communication-aware formation control strategy for dynamical multi-agent systems. *Journal of the Franklin Institute* **2015**, *352*, 3701–3715.

[7] Li, H.; Peng, J.; Zhang, X.; Huang, Z. Flocking of mobile agents using a new interaction model: A cyber-physical perspective. *IEEE Access* **2017**, *5*, 2665–2675.

[8] Eberhart, R.; Kennedy, J. Particle swarm optimization. Proceedings of the IEEE international conference on neural networks. 1995; pp 1942–1948.

[9] Erskine, A.; Joyce, T.; Herrmann, J. M. Stochastic stability of particle swarm optimisation. *Swarm Intelligence* **2017**, *11*, 295–315.

[10] López-Franco, C.; Zepeda, J.; Arana-Daniel, N.; López-Franco, L. Obstacle avoidance using PSO. 2012 9th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE). 2012; pp 1–6.

[11] Chołodowicz, E.; Figurowski, D. Mobile robot path planning with obstacle avoidance using particle swarm optimization. *Pomiary Automatyka Robotyka* **2017**, *21*, 59–68.

[12] Zhang, Y.; Wang, S.; Ji, G.; others A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical problems in engineering* **2015**, *2015*.

[13] Nasrollahy, A. Z.; Javadi, H. H. S. Using Particle Swarm Optimization for Robot Path Planning in Dynamic Environments with Moving Obstacles and Target. 2009 Third UKSim European Symposium on Computer Modeling and Simulation. 2009; pp 60–65.

[14] Li, X.; Xing, K.; Lu, Q. Hybrid particle swarm optimization algorithm for scheduling flexible assembly systems with blocking and deadlock constraints. *Engineering Applications of Artificial Intelligence* **2021**, *105*, 104411.

[15] Pal, N. S.; Sharma, S. Robot path planning using swarm intelligence: A survey. *International Journal of Computer Applications* **2013**, *83*, 5–12.

[16] Chen, T.; Chen, S.; Zhang, K.; Qiu, G.; Li, Q.; Chen, X. A jump point search improved ant colony hybrid optimization algorithm for path planning of mobile robot. *International Journal of Advanced Robotic Systems* **2022**, *19*, 17298806221127953.

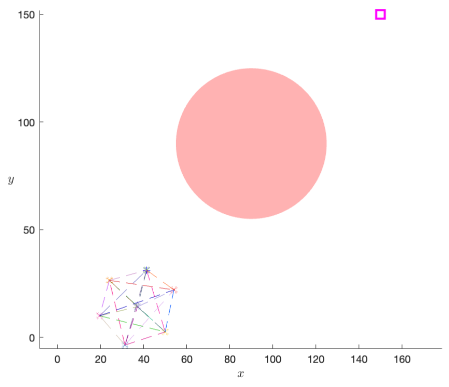
[17] Mandloi, D.; Arya, R.; Verma, A. K. Unmanned aerial vehicle path planning based on A* algorithm and its variants in 3D environment. *International Journal of System Assurance Engineering and Management* **2021**, *12*, 990–1000.

[18] Hussein, A.; Mostafa, H.; Badrel-din, M.; Sultan, O.; Khamis, A. Metaheuristic optimization approach to mobile robot path planning. 2012 International Conference on Engineering and Technology (ICET). 2012; pp 1–6.

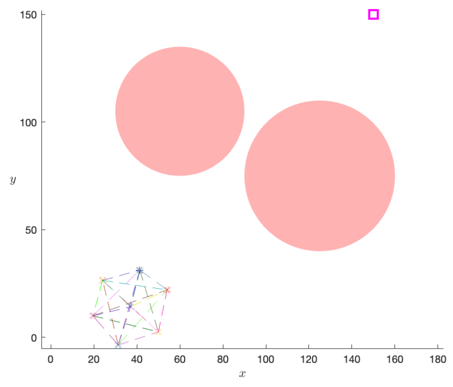
[19] Xie, F.; Müller, M.; Holte, R. Jasper: the art of exploration in greedy best first search. *The Eighth International Planning Competition (IPC-2014)* **2014**, 39–42.

[20] Karur, K.; Sharma, N.; Dharmatti, C.; Siegel, J. E. A Survey of Path Planning Algorithms for Mobile Robots. *Vehicles* **2021**, *3*, 448–468.

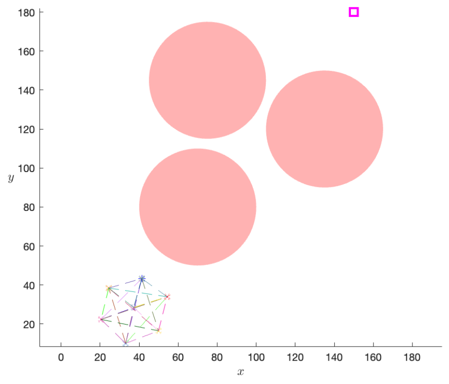
[21] Costa, M. M.; Silva, M. F. A Survey on Path Planning Algorithms for Mobile Robots. 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). 2019; pp 1–7.



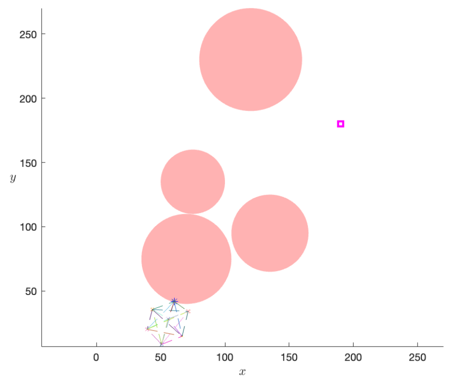
(a) One Jamming Area



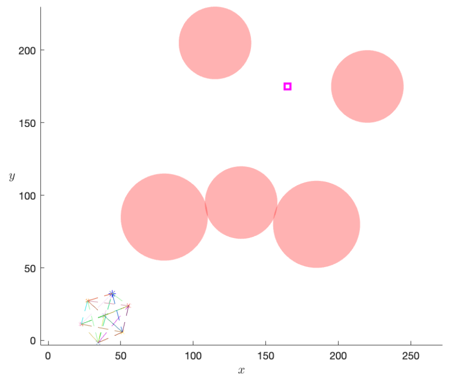
(b) Two Jamming Areas



(c) Three Jamming Areas

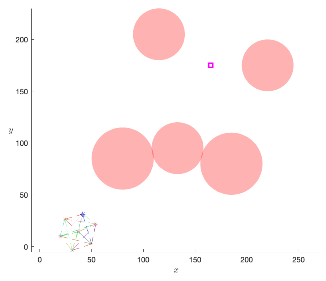


(d) Four Jamming Areas

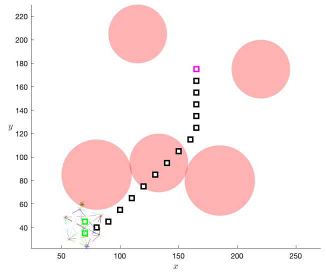


(e) Five Jamming Areas

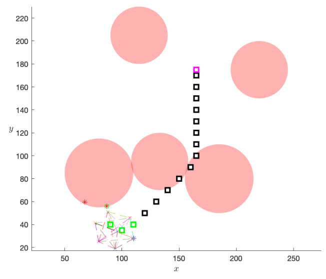
Figure 4: Different simulation environments that were tested.



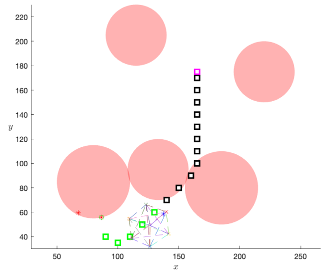
(a) No Jam Points



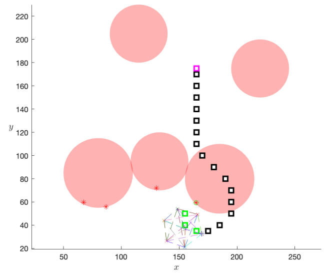
(b) One Jam Point



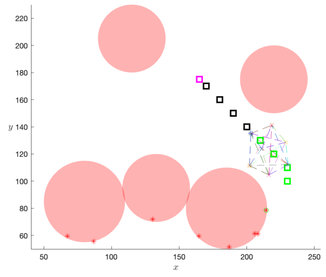
(c) Two Jam Points



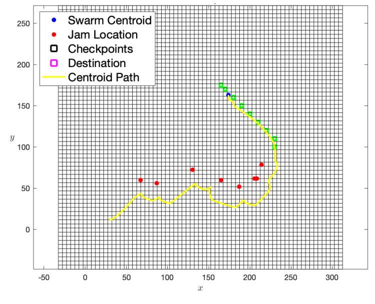
(d) After Two Jam Points



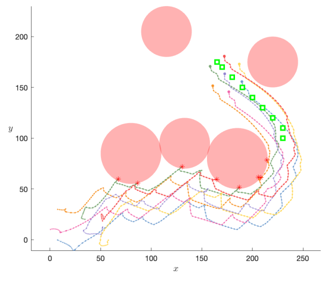
(e) Four Jam Points



(f) Eight Jam Points



(g) Path Planning Grid Map



(h) Agents Trajectory

Figure 5: Example of a simulation run on the five jamming areas environment using A* path planning.

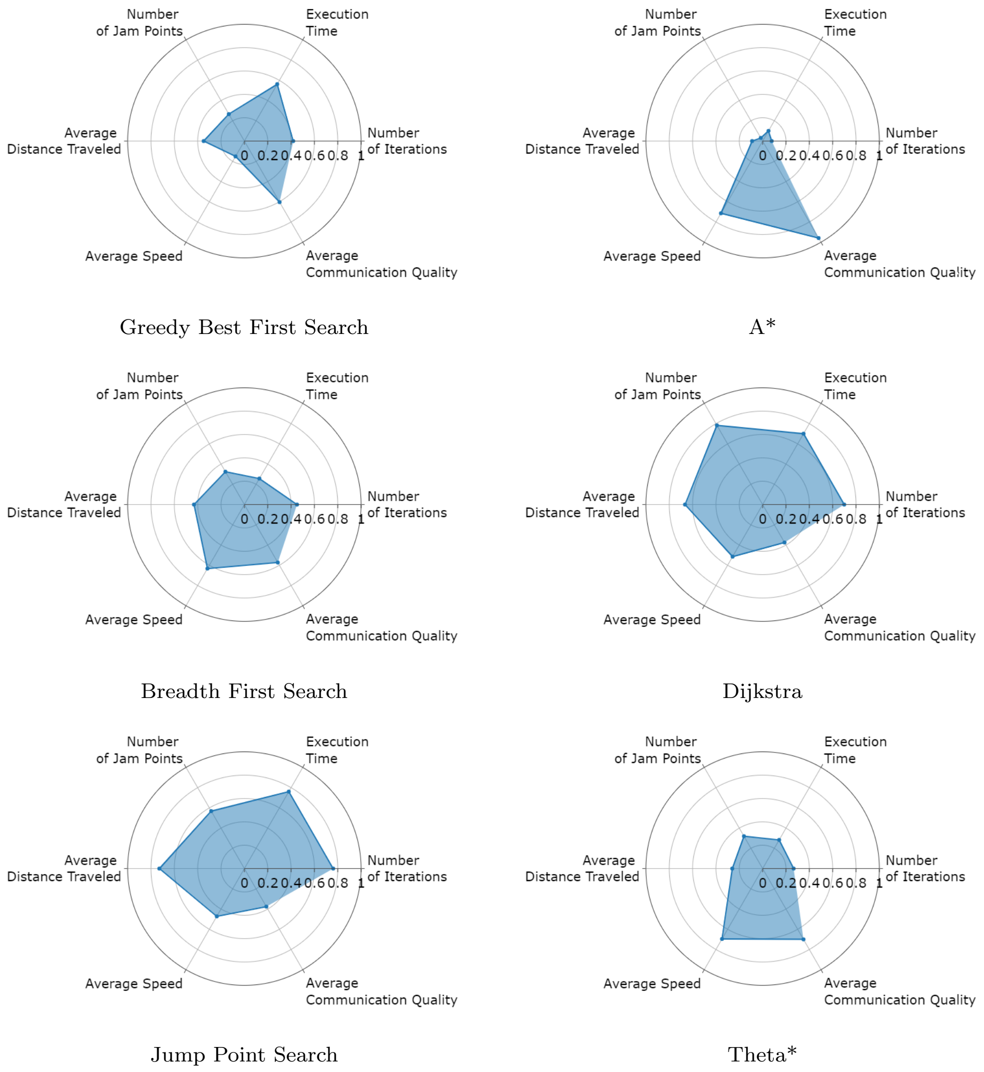


Figure 6: Radar charts representing the performance metrics of different path planning algorithms.