Swarm Intelligence Path-Planning Pipeline and Algorithms for UAVs: Simulation, Analysis and Recommendation

Wyatt Harris[1], Sean Tseng[1], Tabatha Viso[1], Max Weissman[1] and Chun-Kit Ngan[2]
[1]*Department of Robotics Engineering, Worcester Polytechnic Institute, 100 Institute Rd., Worcester, MA, U.S.A.*
[2]*Data Science Program, Worcester Polytechnic Institute, 100 Institute Rd., Worcester, MA, U.S.A.*

Keywords: Artificial Intelligence, Artificial Potential Field, Bio-Inspired Optimization, Heuristic Search, Hybrid Optimization Methods, Path Planning, Policy Gradient, Swarm Robotics, State-of-the-Art, UAV.

Abstract: This research work aims to support domain experts in the selection of proper path planning algorithms for UAVs to solve a domain business problem (i.e., the last mile delivery of goods). In-depth analysis, insight, and recommendations of three promising approaches, including reinforcement learning-based, bio-inspired-based, and physics-based are used to address the multi-agent UAV path planning problem. Specifically, the contributions of this work are fourfold: First, we develop a unified pipeline to implement each approach to conduct this analysis. Second, we build a 2D UAV path planning environment to simulate each approach. Third, using this 2D environment, we run the 450 simulations in three different group sizes of swarm UAV agents (i.e., 3, 5, and 10) within three environments of varying complexity (i.e., Easy, Intermediate, and Hard). We aggregate the simulation data and compare their performance in terms of success rate, run-time, and path length while using the classical A* Search as a baseline. Finally, based upon the performance of each approach and our analytical investigations, we provide informed recommendations for the optimal use case of each UAV path planning approach. The recommendations are presented using parameters for environmental complexity and urgency of goods delivery.

1 INTRODUCTION

Swarm robotics is a field that describes groups of individual agents that can act in a decentralized manner through local decision making processes.

Initially, the field of swarm robotics started as a section of research focused on robot communications. Now the field has grown to house a variety of interdisciplinary applications and domains for both commercial and military sectors (Theraulaz et al., 2021). One of the main advantages of using swarm robotics over individual robotic agents is that swarm robotics may leverage smaller and cheaper robots to accomplish tasks as effectively, or in some cases, more effectively than individual agents. This is especially clear in large environments with high time-cost actions such as search and rescue, path planning to targets, warehouse routines, and military engagements (Tan and Zheng, 2013) (Zhen et al., 2020).

Unmanned Aerial Vehicles (UAVs) as members of robotic swarms have gained popularity in recent years as possible tools for last mile delivery of goods. Last mile delivery, or final mile delivery, is the movement of goods from a warehouse to a customer's house or designated package area (DHL, 2023). This interest stems from companies seeking solutions to higher delivery volumes, aging workforce, and on-time delivery. Several companies such as Alibaba, Alphabet, Amazon, DHL, UPS and even Domino's have experimented with UAV delivery of groceries, medical supplies and mail (Li and Kunze, 2023).

Amazon has developed a UAV targeting a 60 minute delivery time once a customer orders a medication as a part of Amazon Pharmacy's delivery options. The company is also increasing the range of these drones from 5 km to 15 km, reducing the build cost from $ 146,000 to $60,000 and increasing the number of goods available to be shipped via UAVs (Staff, 2023) (Kim and Long, 2022). Another notable commercial use case of UAVs is the company Zipline which effectively delivers important medical supplies across remote and inaccessible regions in Ghana (Li and Kunze, 2023).

However, operational research analyzing the cost per shipment and cost per payload unit between unmanned ground drones against UAVs found that the factors "operator costs per hour" and "average beeline service radius" in combination with "average cruising speed" were the most crucial cost driver variables of delivery. The authors noted that in order to drive

Harris, W., Tseng, S., Viso, T., Weissman, M. and Ngan, C.
Swarm Intelligence Path-Planning Pipeline and Algorithms for UAVs: Simulation, Analysis and Recommendation.
DOI: 10.5220/0012686000003690

In *Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS 2024) - Volume 1*, pages 747-758
ISBN: 978-989-758-692-7; ISSN: 2184-4992

down costs and make UAV's competitive with regular ground vans, the number of drones per human operator must increase (Li and Kunze, 2023).

The demand for UAVs is increasing, as noted in a Statista report, 630,000 units were shipped in 2020 with an expectation of a four-fold increase in 2025 to 2.6 million units (Tractica, 2019). Peak demand is heavily influenced by military and defense spending, with an expectation of the UAV market growing from $25 billion in 2018 to nearly $70 billion in 2029 (BIS, 2019). This demonstrates that interest in UAVs is only growing. In order to make UAV drones more viable, larger groups of drones must be leveraged to drive down costs, increase reliability and ultimately be more effective than traditional vans. Swarm UAVs leverages the growing global interest in UAV drones and can potentially provide more effective last mile delivery.

Multi-agent path planning for UAV drones last mile delivery of goods encompasses three distinct categories of approaches, each offering unique perspectives and methodologies. In reinforcement learning (RL), artificial intelligence principles enable agents, such as UAVs, to dynamically learn optimal paths through trial and error, adapting their decision-making over time based on feedback. Alternatively, bio-inspired algorithms draw inspiration from nature, replicating the decentralized and collaborative behaviors observed in social insects or flocking birds. These approaches excel in promoting adaptability and resilience within the swarm. Lastly, physics-based algorithms ground themselves in principles of physics and motion, modeling interactions and constraints among agents and their environment. By considering factors such as collision avoidance and potential fields, physics-based algorithms provide a foundation for realistic and reliable path planning, particularly in scenarios where precise control and adherence to physical constraints are crucial. Together, these categories contribute diverse tools to address the challenges posed by complex and dynamic environments in swarm multi-agent systems.

The reinforcement learning based algorithm studied in this work is the Multi-Agent Deterministic Policy Gradient (MADPG) for UAV path planning (Zhu et al., 2023). The algorithm is a lighter version of the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) without neural networks to facilitate faster computation speeds. UAV agents will choose an action based on its value and then receive a reward at the next state depending on the performance relative to the goal position. There is an actor and critic system that updates the parameters of all agents based on the swarm's progress (Huang et al., 2020). In the case of last mile deliveries, the algorithm can be used for organizing a swarm of drones from different warehouses to converge to a customer's home which prevents the need for consecutive transport of goods from warehouse to another. This method excels at finding optimal policies to maximize rewards in dynamic environments (Wu et al., 2020b). One of the weaknesses of reinforcement learning algorithms is that they seek to find a balance between exploration and exploitation of the environment. Too much exploration can lead to inefficiencies, and a fixation on exploitation may yield a non-optimal solution (Xue et al., 2023).

The bio-inspired approach studied in this work is the promising Hybrid Simplified Grey Wolf Optimization with Modified Symbiotic Organism Search (HSGWO-MSOS) algorithm, which combines the strengths of two relatively novel bio-inspired optimization algorithms proposed in 2014: Grey Wolf Optimization (GWO) and Symbiotic Organisms Search (SOS). Although this SOTA method was only proposed for single-agent UAV path planning, it offers great potential for multi-agent optimization, since many comparable SOTA methods have successfully utilized GWO for similar applications (Xu et al., 2020). This algorithm is successful because it utilizes the hierarchical structure of GWO to balance exploration and exploitation, while also promoting cooperation and enabling effective local searches through the commensal exploration method from SOS. By combining these approaches, the proposed HSGWO-MSOS has been shown to outperform classical algorithms in UAV path planning, delivering efficient global searches and effective local refinement (Qu et al., 2020). In the context of last mile delivery, this approach would function efficiently by continuously identifying the most optimal route taken by a UAV in a group of delivery agents, leveraging this path to guide all agents, and simultaneously permitting exploration of nearby alternative routes by remaining agents. However, HSGWO-MSOS may struggle with potential slow convergence and scalability issues, which means it may not always find the most optimal delivery routes or may not be well suited for optimizing a large number of delivery agents at a time.

Finally, a successful physics-based approach is Improved Artificial Potential Field (APF) method, which incorporates multiples advancements over the traditional APF method. The algorithm incorporates attractive and repulsive fields between UAV agents to maintain an ideal distance for effective path planning. APF is a widely-used algorithm for collision avoidance in robotics path planning (Wu et al., 2020a). It divides the environment into attractive and repulsive forces, with goals exerting attractive forces and obsta-

cles exerting repulsive forces on the UAV. By calculating these forces and taking their vector sum, the direction of the UAV is determined. The simplicity and efficiency of APF, along with its ability to generate smooth trajectories, have made it a popular choice in various applications, especially in dynamic environments such as UAV delivery services for last mile deliveries. Variations of the base APF algorithm and similar physics-based approaches have been proposed in the UAV delivery and UAV search and rescue domains (Zhao et al., 2020). The base APF algorithm has limitations, such as convergence to local minima and a jitter problem. The SOTA Improved APF method overcomes many, but not all, of these limitations, making it suitable for multi-UAV systems where global optima and swarm compactness are crucial (Zhang et al., 2022). While the Improved APF algorithm has shown promising results in simulation experiments, providing effective collision avoidance and real-time performance, it still needs to maintain a trade-off between speed and performance, and require rigorous tuning of various parameters.

In this paper, we focus on swarm robotics and its application to multi-agent path planning for UAVs in unfamiliar environments and unknown static obstacles. This work aims to support domain experts in the selection of proper path planning algorithms for UAVs to solve a domain business problem (i.e., the last mile delivery of goods). In-depth analysis, insight, and recommendations of three promising approaches, including reinforcement learning-based (i.e., Multi-Agent Deterministic Policy Gradient (MADPG)), bio-inspired-based (i.e., Hybrid Simplified Grey Wolf Optimization with Modified Symbiotic Organism Search (HSGWO-MSOS)), and physics-based (i.e., Improved Artificial Potential Field (APF)) are used to address the multi-agent UAV path planning problem. Specifically, the contributions of this work are fourfold: First, we develop a unified pipeline to implement each approach to conduct this analysis. Second, we build a 2D UAV path planning environment to simulate each approach. Third, using this 2D environment, we run the 450 simulations in three different group sizes of swarm UAV agents (i.e., 3, 5, and 10) within three environments of varying complexity (i.e., Easy, Intermediate, and Hard). We aggregate the simulation data and compare their performance in terms of success rate, run-time, and path length while using the classical A* Search as a baseline. Finally, based upon the performance of each approach and our analytical investigations, we provide informed recommendations for the optimal use case of each UAV path planning approach. The recommendations are presented using parameters for environmental complexity and urgency of goods delivery.

The remainder of the paper is organized as follows: Section 2 describes the development of the simulation environment and data flow. Section 3 explains the pipeline for implementing each approach, outlining the overall methodology, pseudo-code, and an example. Section 4 details the experimental analysis of all approaches within the bounds of our simulation environment. Results of these simulations as well as situational recommendations are included. Finally, the results are summarized and future work is outlined in Section 5.

2 SIMULATION ENVIRONMENT

To implement the three SOTA algorithms as well as the baseline A* Search algorithm, a simulation environment was developed. Pygame, an open source Python library typically used for multimedia applications, was leveraged to design the control environment. The pygame backbone allows for real-time simulations of all methodologies included in this body of work (Shinners, 2011).

The simulation environment was abstracted to three main objects:

1. Agents
2. Obstacles
3. Goal State

Agents, which represent UAVs, are defined with a preset radius, identification number, (x,y) tuple for real time position, and more. Additional parameters are included for tracking agent position over time. Obstacles are static circles defined with a (x,y) tuple as well as radius. Their cylindrical shape ideally represent avoidance areas around trees, poles and tall buildings which UAVs commonly encounter and need to avoid. Trees come in a variety of shapes and widths, and the obstacle class can effectively represent that variance with custom sizes and overlap. The goal state is a simple point with a static (x,y) tuple for position. The superposition of these elements creates unique simulation environments for implementation of any path planning algorithm.

As shown in Figure 1, at simulation start, the user is prompted to input the chosen algorithm, number of agents in the swarm, and the environment difficulty. This defines the problem. The start and end/goal positions of the agents are randomly selected. Then, the inputs are used in the define algorithm, swarm size and environment stage. The objects for all agents are created along with instances of the three algorithms. Then swarm sizes of 3, 5, and 10 are created. Pygame

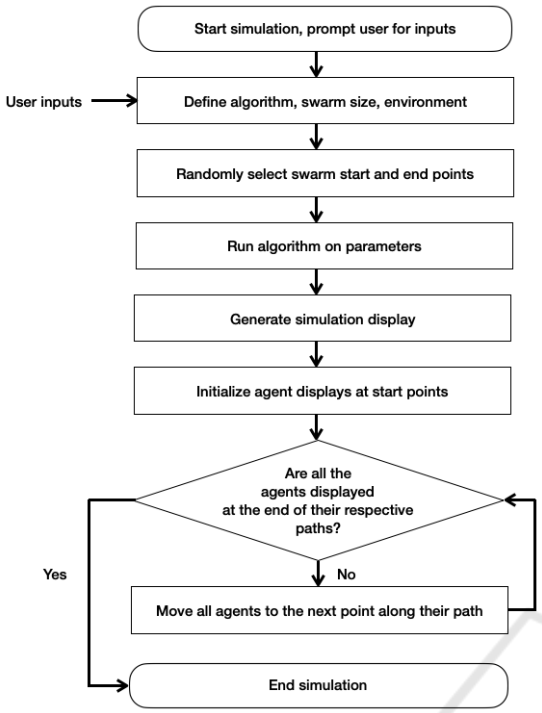

Figure 1: Simulation environment Workflow.

is used to create a simulation display with the environment obstacles. Next, the program generates random initial start positions as well as the goal position for all agents. All algorithms are run for all agents in the run algorithms stage. The simulation display is created to showcase the agents' results which is followed by plotting the agents' initial positions. There is a loop indicated at the diamond stage where every agent in the group completes their movement until they have reached the end of path planning. Lastly, once the iteration count or goal position is reached, the simulation is terminated.

While the structure of the environment can be randomly generated, predetermined environments were used to directly compare the performance of algorithms. Three obstacle configurations of increasing difficulty were developed, and are shown in Figure 2. The easy environment represents eight equal obstacles of small diameter. In a last mile delivery application, these obstacles might represent small trees, lampposts, or other small structures, possibly organized in a structured way. The intermediate environment is more complex featuring a 5x5 grid of obstacles of varying radii. Some cells in this environment are left empty to give the UAVs ample room to navigate. This environment represents a slightly more complex last mile delivery scenario, where there might be a combination of trees of varying sizes and infrastructure such as an electrical tower or lampposts. The hard environment is also on a 5x5 grid, but some obstacles are significantly larger than in the intermediate environment. The lack of free space in the hard environment makes it more difficult for the UAVs to traverse without collisions. This represents a tough delivery scenario with trees, infrastructure, and other structures such as water towers, silos, or windmills.

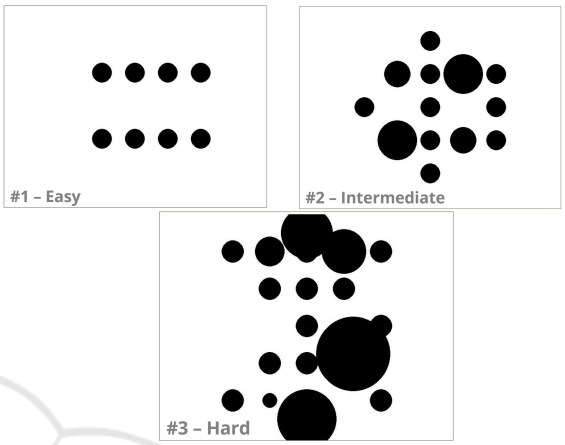

Figure 2: Simulation environments of increasing complexity.

Using the developed environments, the simulation was run for each algorithm, environment, and number of agents (3, 5, and 10). Each of these unique configurations was run 50 times. For example, the MADPG algorithm with five agents was run in the hard environment 50 times. In total, each algorithm was run 450 times. The data collected was formatted and analyzed to determine the characteristics of the implemented algorithms.

3 PROPOSED PIPELINE AND ALGORITHM IMPLEMENTATION

The three different algorithms will have their own classes in their own files and the main RunSim script will call upon those classes to implement the different methods. A* search algorithm will also be simulated for each iteration of the script. Altogether, the algorithms were run 450 times and produced results used for analysis in Section 4.

In Figure 3, the user determines inputs such as number of agents and which environment. This feeds the script to run the algorithms with those parameters.The agents' display is generated and then the agents enter the loop where they keep moving until

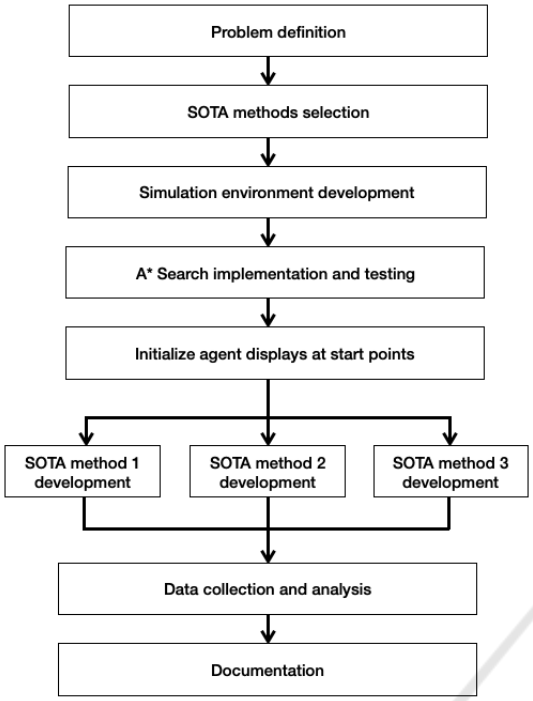

Figure 3: Proposed pipeline of simulation and general flow logic.

all of them reach the goal or the simulation time expires. Lastly, the data from the simulations is saved and compiled for analysis.

3.1 MADPG Algorithm

3.1.1 Approach

The MADPG algorithm iterates through all agents' set of policies which themselves contain five states. Afterwards, the policies are evaluated based on the value function Bellman equation from Markov Decision Process. The gradient of these results are calculated and then compared. If a lower gradient is detected, the main critic will push an update to all actors or agents to become more forgiving of lower reward policies, with the opposite case leading to stricter thresholds for acceptable polices.

Inside each agent are a couple methods, including state validator, action, reward and critic update functions. The action method generates a policy based on the simulated annealing search algorithm, where the agents have a certain initial searching temperature that is cooled over time as the agents approach the goal position. Each move is determined through calculating the probability of a future move bringing the agent closer to the goal. Each navigation episode consists of five transitions, which are the transitions from five different states or in this case positions. The algorithm checks if the goal is reached within the predefined movement speed; if so, the current position is set to the goal position. Otherwise, the algorithm adds a random angle between -30 to 30 degrees to generate new potential destinations en route to the goal.

The policy path is evaluated for validity in the state space. If the policy is acceptable with respect to the boundaries according to the validator, the algorithm then evaluates the energy cost for that transition using the "reward" function. This is then used to calculate the probability for approaching the goal state using the aforementioned thermal equilibrium probability equation. The reward function calculates the agent's reward for a given state based on the maximum possible distance from the target position subtracted by the current Euclidean distance to the goal. There are any obstacles nearby within 12 units, a penalty is applied to reward through subtraction from the reward. This means the higher the reward, the closer the agents are to the goal, while a lower number indicates a large distance to the goal and encountering many obstacles.

The critic update method takes signals from the overarching critic to then update the parameters of the agents. As agents complete the goal, the remaining agents will be retrained differently as the algorithm continuously updates the parameters. This is to simulate encouragement from other completed agents.

3.1.2 Pseudo-Code

First, obstacles, MADPG agent object, discount_factor, gradient, and paths must be initialized. The positions, sizes and quantity of obstacles are determined by the script. This will be used by agents to determine if collisions happen. The discount_factor is hard coded in the program; it is a decimal coefficient that changes the balance between exploration of the environment against exploitation of gathered information. Gradient and paths are container lists for future data regarding the agents' paths and performance.

The action method is an epsilon greedy method that uses simulated annealing to enable random exploration. Epsilon greedy finds the highest reward action and proceeds with it for five steps. The simulated annealing probability triggers a random action and helps with exploration. For agent_next_reward, the method searches the whole list of state-action values for the agent and finds the one with largest reward.

In Step 1, there is a loop for each agent to call the action method and move position. The outputs of this are agents' paths, path length and reward. In Step 2, the value of each agent's value function is calculated according to the reward and argmax of a future action

Initialize Obstacles, MADPG agent object, discount_factor, gradient, paths, goal
Methods
action: Moves agent using epsilon greedy and simulated annealing decision making
agent_next_reward: Calculates max of agent's next reward
for *each episode* **do**
  for *each agent* **do**
    (1) Call action method for five iterations and store path, length
    (2) Calculate reward from current state against obstacles object, distance to goal
    (3) $V(S) = \text{reward} + \text{discount\_factor} \times \text{agent\_next\_reward}$
  end
  (4) Calculate gradient for given state and append to local memory
  (5) Compare gradient to change critic networks of all agents
  (6) Append state to agent memory for plotting
  (7) Append local path data to expected path list for plotting
  if *All agents reach goal* **then**
    End loop
  end
end
return *Executed paths for all agents*

Algorithm 1: Reinforcement Learning Approach: MADPG Pseudo Code.

reward, the discount value was set to be 0.5 to balance exploration and exploitation. The gradient of all the agents' reward functions is taken for the current iteration in Step 3, which is simply the slope of current reward values from the previous episodes. This is then used to inform Step 4, which is to compare the change in the gradient and then subsequently send an update to all agents. If the gradient becomes negative, an update is sent to make the agents more exploratory and if the gradient is positive the update makes the agents more focused. This tuning is accomplished via increasing and decreasing the threshold of accepting higher or lower quality values. Lastly in Step 5, the state history of each agent is appended to help with plotting. Steps 6, and 7 store the state and path history of the agents for future analysis. The final if statement ends the episode loop once all agents reach the goal position.

For the example of five UAV delivery drones in the swarm, each UAV will start at different random positions, representing different warehouses or distribution centers. The exploratory initial policy and similar reward values near the initial starting position guides them to make exploratory moves in the environment. As the simulation iterates, the collective states from the past five iterations are stored and analyzed for updating the critic. The environment has obstacles and some agents will enter the nearby boundary radius of those regions. This triggers punitive rewards to disincentive the agents. Instead agents pick locations that circumvent obstacle and gain higher rewards the closer they are to the goal. The episode ends after the specified number of iterations are complete or after all UAV agents reach the delivery goal.

3.2 HSGWO-MSOS Algorithm

3.2.1 Approach

As part of a broader agent-based simulation framework, the Wolf class inherits from the Agent class, allowing it to benefit from generic functionalities shared among different agents in the simulation. However, each Wolf object is also initialized with additional crucial attributes such as fitness, which represents the fitness of the Wolf's current position in the optimization process, and is-alpha, a boolean flag designating whether the agent is responsible for guiding the optimization process.

Moreover, the class includes two heuristic functions that calculate Euclidean distances between points in the search space and enable agents to navigate and evaluate distances effectively. The update_fitness method plays a vital role in evaluating the fitness of the Wolf's current position in the optimization process. It combines the Euclidean distance to the goal (J_fuel) and the threat posed by nearby obstacles (J_threat) into a weighted fitness value (J_cost). The update_position method governs the updating of the Wolf's position based on its designation, either Alpha or Omega. If the Wolf is an Alpha, it moves towards the goal by normalizing the direction vector, while Omega Wolves adjust their positions based on a strength value that determines the attraction towards the Alpha Wolf. The new position is verified for validity before updating, and the Wolf's path and temporary path are recorded accordingly.

During the exploration phase, commensal agents utilize the explore method to randomly explore the search space by moving in different directions. This method generates a random angle and calculates a new destination point, attempting to strike a balance between exploration and obstacle avoidance while remaining in a commensal state. The make-alpha, make-omega, make-commensal, and i-already-explored methods manage the designation of the

Wolf, allowing it to take on different roles as Alpha, Omega, or commensal explorer, depending on the optimization stage.

3.2.2 Pseudo-Code

Initialize Obstacles, Wolf agent object, Alpha strength factor (constant), goal position, paths
Methods
update_hierarchy Evaluates and compares the fitness of all agents, designates Alpha, Omegas, and commensal explorers
evaluate_fitness Combines the Euclidean distance to the goal (J_{fuel}) and the threat posed by nearby Obstacles (J_{threat}) into a weighted fitness value (J_{cost})
update_position Updates agent position based on Alpha/Omega designation and Alpha strength factor
for *each iteration* **do**
  (1) Update Wolf hierarchy **for** *each Wolf* **do**
    (2) Update agent position
    if *Wolf is a commensal explorer* **then**
      (3) Evaluate 5 random nearby locations
      if *Explored location is better than current position* **then**
        (4) Update agent position
      end
    end
    ;
    (5) Append new position to paths
  end
end
return *Executed paths for all agents*

Algorithm 2: Bio-Inspired Approach: HSGWO-MSOS Pseudo Code.

After initializing the multi-agent HSGWO-MSOS swarm, the first step of the algorithm is to define the fitness of all Wolves and create a hierarchy by identifying the Alpha (best individual), Omegas (all other agents), and commensal explorers (randomly selected Omega Wolves). At each iteration, the Alpha Wolf evaluates (Step 2) the next best position based on its position, nearby obstacles, and the desired goal. Each Omega Wolf consequently evaluates their next best position based on the Alpha's location. In Step 3, the commensal explorer randomly searches five nearby locations and evaluates the potential fitness of these positions. If it finds a location that is better than its current position, it moves to this new spot (Step 4). Otherwise it remains in its current position. After each wolf has updated its position, it appends the new position to its path (Step 5). This process is iteratively repeated from the beginning (Step 1), where the fitness of all Wolves is evaluated once again and a new hierarchy is defined. The agent with the best fitness is defined at the new Alpha. All other Wolves are labeled as Omegas and a new commensal explorer is chosen for the next iteration.

For example, in a swarm of five UAV agents, each UAV's fitness is evaluated at initialization with respect to the goal position and nearby obstacles. The best scoring UAV is designated as the Alpha, while the remaining four UAV are designated as Omegas. One of the Omegas is randomly selected to be the commensal explorer. The Alpha UAV updates its position first, by moving towards the goal position along the normalized direction vector. The four Omega UAVs then update their positions by taking into consideration both the Alpha's new position, as well as a constant strength factor. Finally, the commensal explorer UAV considers five randomly chosen nearby locations and evaluates them. If any of those locations is better than its current position, it moves to it. After all five agents have moved to their positions, the fitness of each UAV is re-evaluated and the hierarchy is updated. The UAV with the best scoring fitness is now considered the Alpha, while the rest are considered Omegas, and a new commensal exploring Omega is also selected. This process is repeated until the algorithm returns the completed paths for the UAV.

3.3 Improved APF Algorithm

3.3.1 Approach

The Improved APF algorithm provides a robust and adaptive framework for modeling multi-UAV systems. While there are multiple iterations of Improved APF algorithms, this work will focus on leveraging SOTA methodology described in Dongcheng (2020) and Zhang (2022). In these methods, there is a common focus of modeling interactive repulsive forces between UAVs.

To properly implement this algorithm, all representative forces are considered for each agent, individually. These forces are derived from the attractive and repulsive potential fields throughout the environment. The attractive force created by the goal counteracts the repulsive forces acting on each agent. Repulsive forces for each obstacle in the environment are implemented as well as repulsive forces between each agent. For each agent, the total force is calculated and broken down into component form, then later vector summed together in order to move the agent forward at each iteration.

It is necessary to include a solution to the jitter problem as an aspect of the Improved APF implementation (Dongcheng and Jiyang, 2020). For each step in the simulation, each agent is tested to analyze if it is in a state of jitter. More specifically, this jitter state occurs when the change in the angle of the resultant force is between 180° and a specified threshold value between the current state and the previous state. If this jitter condition is met, a dynamic step adjustment is included when calculating the next position of each agent. The dynamic step adjustment allows for each agent to alter its step size to smoothly escape from jitter scenarios.

A simplified approach for the Improved APF algorithm is included in the Section 3.3.2 Pseudo-code.

3.3.2 Pseudo-Code

Initialize Obstacle set O, UAV agent set A, Goal State g, paths
Methods
dynamicStepAdjustment: Adjusts step size
updatePosition: Updates the position of the agent
for *each agent a in A* **do**
  (1) Calculate attractive force $F_{att}(a,g)$
  (2) Calculate repulsive force $F_{rep}(a,O)$
  (3) Calculate repulsive force $F_{rep}(a,A)$ from each other agent in A
  (4) Calculate $F_{tot} = \sum F_{att} + \sum F_{rep}$
  if *agent is in jitter state* **then**
    (5) *dynamicStepAdjustment*
  end
  else
    (6) *updatePosition* of a in direction of F_{tot}
  end
  ;
  (7) Append new position to paths
end
return *Executed paths for all agents*

Algorithm 3: Physics-Based Approach: Improved APF Pseudo Code.

To begin, the obstacle set, goal state, path variables, and agent sets are initialized. All agents in the multi-agent swarm are configured to run the Improved APF algorithm. Inside each agent are a set of methods necessary to execute the algorithm. Methods for calculating the attractive force between the agent and the goal, $F_{att}(a,g)$, the repulsive force between the agent and obstacles, $F_{rep}(a,O)$, and the repulsive force between the agents and all other agents in the swarm, $F_{rep}(a,A)$. Another method is used to calculate the total force, F_{tot}, from all acting forces. A dynamic step adjustment method is embedded in this implementation to solve assist the agent in escaping a jitter state. Lastly, an update position function is included to update the position of the agent at the end of each loop. This position is appended to the agent's path for plotting within the simulation. For each agent in the swarm, Steps 1-3 are used to calculate the component forces acting on the agent. Step 4 takes the vector sum of these forces to calculate the total force. A check is made to ensure the agent is not in a jitter state. If the agent falls into a jitter state, Step 5 is executed using the dynamic step adjustment function. If the agent is not in jitter state, Step 6 is executed using the update position function.

Take a five-agent swarm, where all agents are running the Improved APF algorithm. They start in five different positions, and need to avoid obstacles as well as each other to reach the goal state. The pseudo-code defined in Algorithm 3, describes the behavior of these agents at each iteration of the simulation. The algorithm is run until all agents reach the goal state or the iteration limit is reached. At the first iteration, each agent calculates the resultant force by first finding all attractive and repulsive forces acting on the agent. The agent updates its position with regard to this resultant force. The positions of each other agent are paramount to calculating the new position of the current agent, as those repulsive forces affect the resultant force for the given agent. In this example, an assumption is made that the agent closest to the goal gets stuck in a local minima near two obstacles. The subsequent agents behind the furthest agent are able to "push" the stuck agent out of the local minima, allowing for it to reach the goal. Each agent in this swarm will help guide the further agent out of these local minima. It is, however, possible for the last agent to get stuck in this same local minima. If this local minima causes the agent to jitter, the improved APF algorithm leverages dynamic step adjustments to guide the last agent to the goal.

4 SIMULATION RESULTS, ANALYSIS, AND RECOMMENDATIONS

4.1 Success Rate Results

Figure 4 shows the success rate results for each algorithm in each environment. After analyzing the results, MADPG stands out with a high success rate in complex environments. The randomization in the algorithm leads to a high probability of success as the

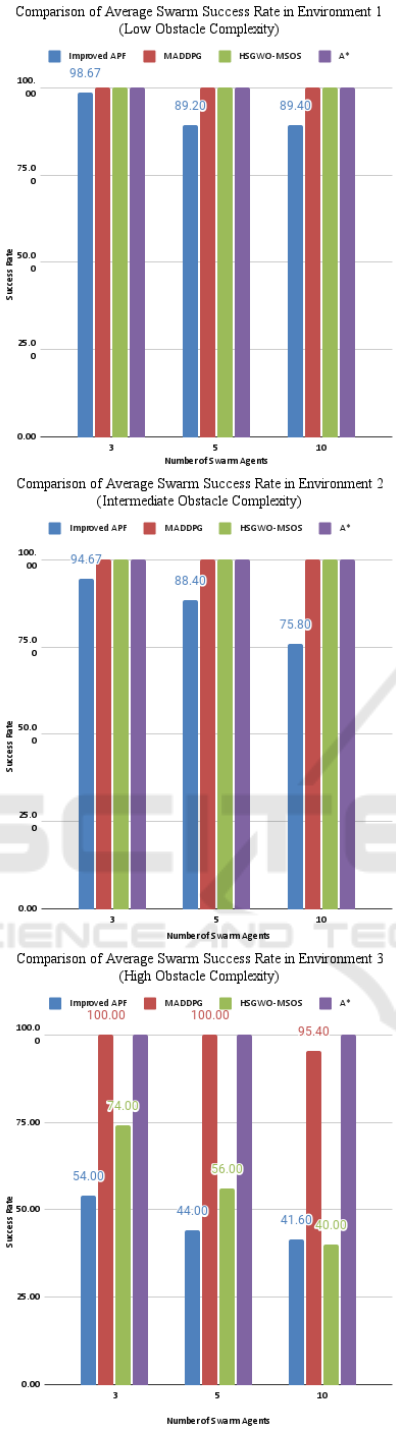

Figure 4: Average swarm success per environment.

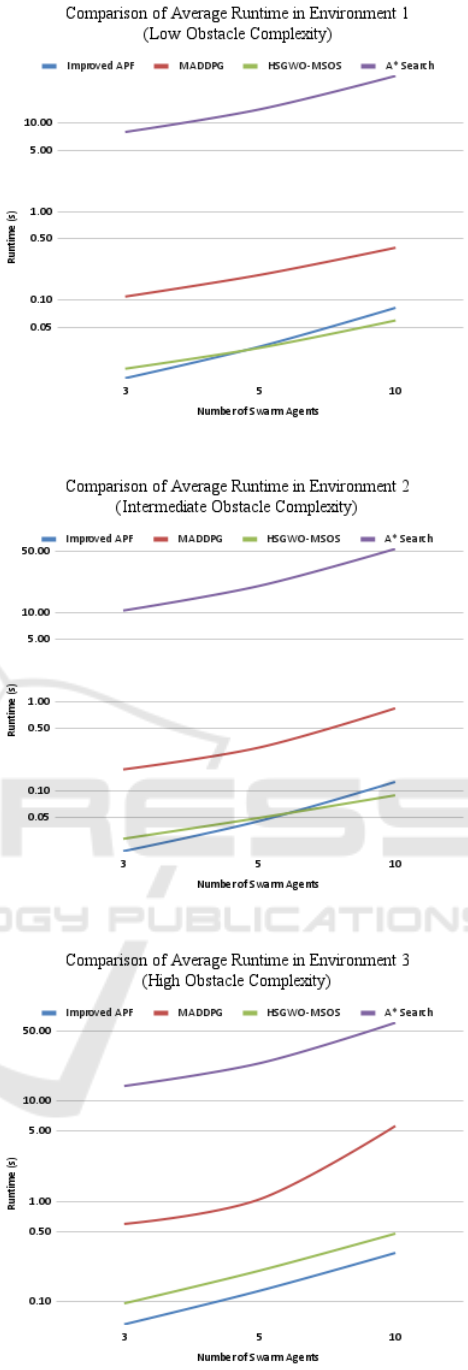

Figure 5: Average algorithm run-time per environment.

number of iterations increases. HSGWO-MSOS and Improved APF seem to have lower success rate, as they are less reliant on randomization to escape local minima, and may become stuck. Improved APF has lower success rates than HSGWO-MSOS due to the likelihood of some agents getting stuck to help other agents. MADPG by far is the best algorithm developed if the goal is to solely increase success rate. This is not considering A* Search, which has a perfect success rate across the environments due to the informed nature of the algorithm.

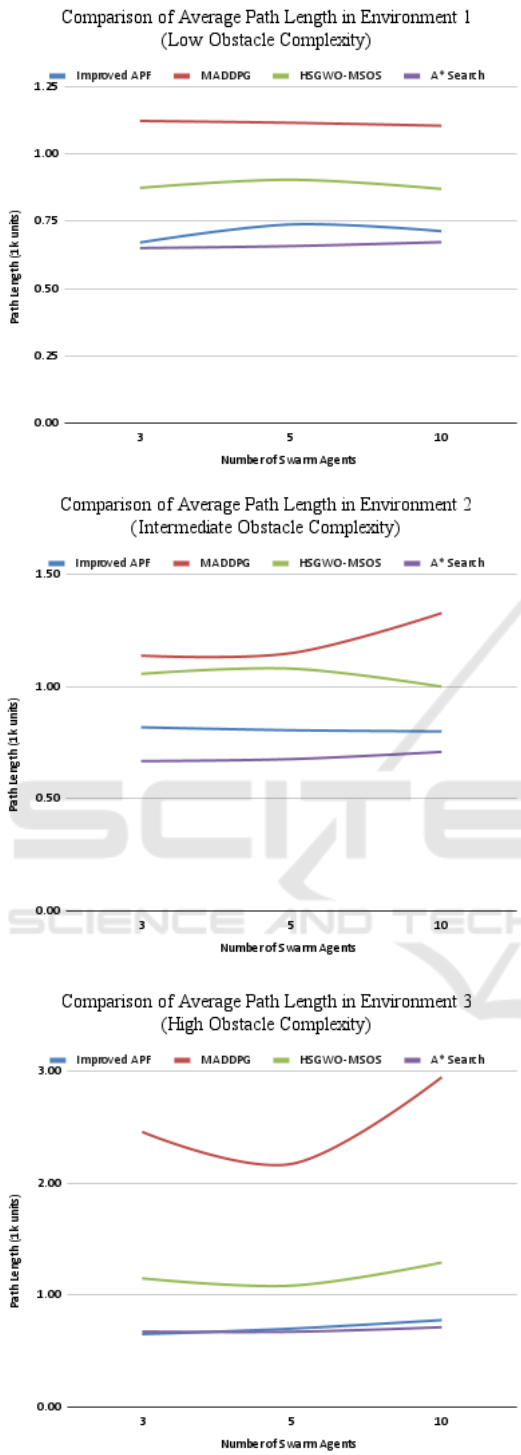

Figure 6: Average algorithm path length per environment.

4.2 Runtime Results

Figure 5 shows the runtime results for each algorithm in each environment. The runtime of Improved APF and HSGWO-MSOS are lower than the runtime of MADPG by a fair margin. This is exasperated in high complexity, large swarm situations. When Improved APF and HSGWO-MSOS agents end up stuck in local minima, an iteration cap may end the program as they are not as reliant on randomization to escape local minima. The MADPG algorithm will sacrifice runtime to escape a local minima using randomization. Improved APF and HSGWO-MSOS sacrifice success rate for a decreased runtime. A* Search in all three environments is a scale of magnitude greater than the runtime of the three SOTA algorithms.

4.3 Path Length Results

Figure 7 shows the path length results for each algorithm in each environment. The average generated path lengths of the algorithms are compared against A* as a baseline. The A* Search algorithm conducts an informed search on the environment instead of a partially informed search, and in turn has a perfect success rate, increased runtime, and also finds the shortest possible path from the start point to the end point. Paths that ended up being incomplete were not included in averages. Improved APF generated path lengths that were very similar to A*, especially in high complexity environments. However, this is partially due to selection bias. As Improved APF's success rate decreases, the path lengths that are generated are more likely to be simple, straight paths to the end goal. HSGWO-MSOS also generated path lengths similar to that of A* and Improved APF. It also was affected by this selection bias of results. MADPG maintained a high success rate across all environments, and was not subject to this selection bias. In high complexity environments, the algorithm created path lengths over three times the length of other algorithms. Even in low complexity environments, it produced the longest path lengths of all the algorithms. In Environment 3, the increased difference of path lengths is likely related to selection bias, but the results of Environment 1 do show that MADPG generated the longest paths, Improved APF generated the shortest paths, and HSGWO-MSOS generated the middlemost paths.

4.4 Recommendations

The code effectively implemented all three SOTA methods: MADPG, HSGWO-MSOS, and Improved APF. Our work completed development of a 2D simulation environment that effectively tested these methods. In addition, a data pipeline was designed to effectively run algorithms for path planning of multi-agent swarms. The average results for each method

are summarized in Table 1.

Table 1: Average results by swarm intelligence path-planning method.

Approach	Classical	RL	Bio-Inspired	Physics-Based
Algorithm	A* Search	MADPG	HSGWO-MSOS	Improved APF
Swarm Success Rate [%]	100.0	99.8	88.4	48.4
Runtime [seconds]	26.67	1.03	0.11	0.09
Path Length [1k units]	677.5	2725.8	1034.6	742.1

MADPG exhibits a high average swarm success rate regardless of swarm size or environment complexity. It is an ideal choice for longer distance deliveries where shipping completion and reliability is paramount. Given that its runtimes and paths are longer on average than the other methods, this reinforcement learning approach is more suitable for scenarios where real-time decision-making or efficient path lengths is not a strict requirement. Alternatively, HSGWO-MSOS provides a reasonable average swarm success rate with lower runtimes and path lengths. This bio-inspired method provides a good balance of success rate and efficiency for scenarios where reliable, timely responses are needed and resources like UAV battery-life may be limited. It would most useful in urban last mile deliveries. Lastly, Improved APF has a lower success rate, but the fastest runtimes and shortest path lengths on average, making this physics-based approach suitable for scenarios where package recovery is more easily attainable, and minimizing distances and rapid decision-making take precedence.

Table 2 summarizes the findings for each algorithm in each environment with respect to the urgency of transported goods. The urgency of goods can be directly correlated to the path length and runtime of the algorithms. Since Improved APF and HSGWO-MSOS exhibit faster runtimes and overall shorter path lengths, they are more suitable for delivery of urgent goods. However, for non-urgent goods, reliability may be prioritized over speed. In this scenario, MADPG is recommended due to its high reliability in complex environments. Figure 7 shows completed simulations and paths for five-agent swarms. Each algorithm is showcased in its optimal and recommended environment.

Table 2: Recommended Algorithm depending on environment complexity and urgency of goods.

Environ.	Easy	Intermediate	Hard
Urgent Goods	Improved APF	HSGWO-MSOS	HSGWO-MSOS
Non-urgent Goods	HSGWO-MSOS	MADPG	MADPG

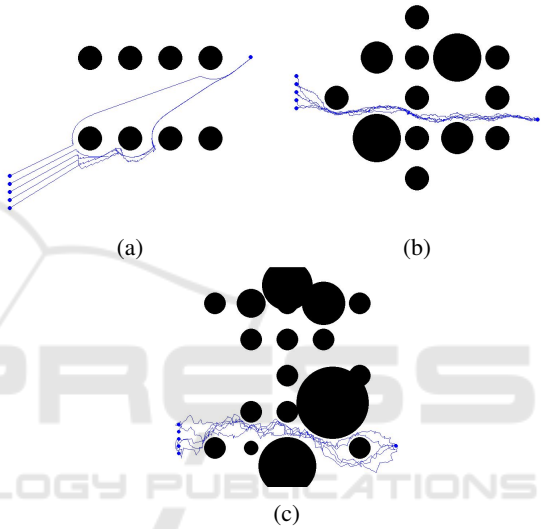

Figure 7: Sample algorithm performance in recommended Environments.
(a) Easy environment with Improved APF.
(b) Intermediate environment with HSGWO-MSOS.
(c) Hard environment with MADPG.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we focus on swarm robotics and its application to multi-agent path planning for UAVs in unfamiliar environments and unknown static obstacles. This work aims to support domain experts in the selection of proper path planning algorithms for UAVs to solve a domain business problem (i.e., the last mile delivery of goods). In-depth analysis, insight, and recommendations of three promising approaches, including reinforcement learning-based (i.e., Multi-Agent Deterministic Policy Gradient (MADPG)), bio-inspired-based (i.e., Hybrid Simplified Grey Wolf Optimization with Modi-

fied Symbiotic Organism Search (HSGWO-MSOS)), and physics-based (i.e., Improved Artificial Potential Field (APF)) are used to address the multi-agent UAV path planning problem. Specifically, the contributions of this work are fourfold: First, we develop a unified pipeline to implement each approach to conduct this analysis. Second, we build a 2D UAV path planning environment to simulate each approach. Third, using this 2D environment, we run the 450 simulations in three different group sizes of swarm UAV agents (i.e., 3, 5, and 10) within three environments of varying complexity (i.e., Easy, Intermediate, and Hard). We aggregate the simulation data and compare their performance in terms of success rate, run-time, and path length while using the classical A* Search as a baseline. Finally, based upon the performance of each approach and our analytical investigations, we provide informed recommendations for the optimal use case of each UAV path planning approach. The recommendations are presented using parameters for environmental complexity and urgency of goods delivery. While these recommendations are relevant in the domain of last mile delivery, further research is needed to investigate elements of the problem not covered in this work. Further research includes, but is not limited to, inter-agent collisions, dynamic obstacles, 3D path finding, and novel algorithms for solving multi-agent path planning more efficiently and effectively.

ACKNOWLEDGEMENTS

The authors would like to acknowledge and thank Professor Chun-Kit Ngan for his support and advice during this project.

REFERENCES

BIS (2019). Global uav market value in 2018 and 2029 (in billion u.s. dollars). Technical report, Statista.

DHL (2023). 4 ways to improve your last-mile delivery performance. *DHL*.

Dongcheng, L. and Jiyang, D. (2020). Research on multi-uav path planning and obstacle avoidance based on improved artificial potential field method. *2020 3rd International Conference on Mechatronics, Robotics and Automation (ICMRA)*.

Huang, Y., Wu, S., Mu, Z., Long, X., Chu, S., and Zhao, G. (2020). A multi-agent reinforcement learning method for swarm robots in space collaborative exploration. *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*.

Kim, E. and Long, K. (2022). Amazon will pay a whopping 63 dollars per package for drone delivery in 2025 and it shows just how the company is still grappling with cost issues for last-mile delivery. *Business Insider*.

Li, F. and Kunze, O. (2023). A comparative review of air drones (uavs) and delivery bots (sugvs) for automated last mile home delivery. *Logistics*.

Qu, C., Gai, W., Zhang, J., and Zhong, M. (2020). A novel hybrid grey wolf optimizer algorithm for unmanned aerial vehicle (uav) path planning. *Knowledge-Based Systems, 194, 105530*.

Shinners, P. (2011). Pygame. http://pygame.org/.

Staff, A. (2023). Amazon announces 8 innovations to better deliver for customers, support employees, and give back to communities around the world. Technical report, Amazon.

Tan, Y. and Zheng, Z. (2013). Research advance in swarm robotics. *Defence Technology*.

Theraulaz, G., Dorigo, M., and Trianni, V. (2021). Swarm robotics: Past, present, and future. In *Proceedings of the IEEE*.

Tractica (2019). Projected commercial drone hardware unit shipments worldwide from 2020 to 2025. Technical report, Statista.

Wu, E., Sun, Y., Huang, J., Zhang, C., and Li, Z. (2020a). Multi uav cluster control method based on virtual core in improved artificial potential field. *IEEE Access, 8, 131647–131661*.

Wu, S., Huang, Y., Mu, Z., Long, X., Chu, S., and Zhao, G. (2020b). A multi-agent reinforcement learning method for swarm robots in space collaborative exploration. *6th International Conference on Control, Automation and Robotics*.

Xu, C., Xu, M., and Yin, C. (2020). Optimized multi-uav cooperative path planning under the complex confrontation environment. *Computer Communications, 162, 196–203*.

Xue, J., Zhu, J., Du, J., Kang, W., and Xiao, J. (2023). Dynamic path planning for multiple uavs with incomplete information. *Electronics, 12(4), 980.*

Zhang, M., Liu, C., Wang, P., Yu, J., and Yuan, Q. (2022). Uav swarm real-time path planning algorithm based on improved artificial potential field method. *2021 International Conference on Autonomous Unmanned Systems (ICAUS 2021)*.

Zhao, Y., Liu, K., Lu, G., Hu, Y., and Yuan, S. (2020). Path planning of uav delivery based on improved apf-rrt* algorithm. *Journal of Physics: Conference Series*, 1624:042004.

Zhen, Z., Chen, Y., Wen, L., and Han, B. (2020). An intelligent cooperative mission planning scheme of uav swarm in uncertain dynamic environment. *Aerospace Science and Technology, 100, 105826*.

Zhu, J., Xue, J., Du, J., Kang, W., and Xiao, J. (2023). Dynamic path planning for multiple uavs with incomplete information. *Electronics*.