EAI Endorsed Transactions
on AI and Robotics
Research Article EAI.EU

Methodologies enabling Mobile Robots Collective Motion: A Comprehensive Review

Maikano Kenneth Oganne[1], Thabo Semong[1,*], Dimane Mpoeleng[1]

[1]Department of Computing & Informatics, School of Pure and Applied Sciences, Botswana International University of Science and Technology (BIUST), Private Bag 16, Palapye, Botswana

Abstract

The demand for multi-robotic systems continues to grow. As a result, there is a notable interest among researchers and industry experts to develop control methods or policies to determine how multi-robot system members should cooperate in order to enforce cohesion. These methods can be classified as nature-inspired, self-propelled particles (SPP) based, and learning-based algorithms. This paper presents a comprehensive review of these methods. The main aim is to identify, analyze and discuss the strengths and weaknesses of these methods. Additionally, this paper aims to suggest the optimal control method for enabling effective collective motion of multiple robots and to highlight the identified research gaps and suggest how they can be addressed. Nature inspired algorithms such as ant colony optimization (ACO) are simple and easy to implement when compared to others. However, they require careful parameter tuning for them to operate optimally. On the other hand, self-propelled particles (SPP) based algorithms are decentralized, easy to configure, and produce naturalistic emergent behavior, but suffer from high oscillation when experiencing inaccurate sensor readings and communication delays. Addressing the limitations of nature-inspired and SPP based algorithms should focus on developing control methods with learning abilities. Although learning-based methods are computationally intensive, they are capable of handling sensor inaccuracies and communication latency, making them well suited for the collective motion requirements of mobile robots, particularly in highly dynamic environments. Various strategies, including network pruning, the use of TinyML, and Central Training with Distributed Execution (CTDE), can be employed to optimize learning-based methods for robots with limited resources.

Received on 12 February 2026; accepted on 04 June 2026; published on 15 June 2026

Keywords: Reinforcement Learning, Mobile Robots Collective Motion, Nature Inspired Algorithms, Self Propelled Particles, SPP algorithms, Learning Based Methods

doi:10.4108/airo.11922

1. Introduction

Mobile Robots Collective Intelligence is the art of coordinating multiple mobile robots to work towards a goal cohesively [1, 2]. Multiple well harmonized robots provides the means to execute sophisticated tasks that are arduous to perform with a single robot. Such tasks may include reconnaissance, mapping and many more [3–5]. Well coordinated Robots, normally with limited resources, presents an interesting emergent problem solving capability perceived in natural systems such as fish schools or shoals. Arkin et al. [6] acknowledge that a group of robots with cognitive abilities can decompose and distribute tasks amongst themselves (distributed actions), and perform different tasks at the same time (inherent parallelism). Another inherent advantage is that a system of multiple robots may exhibit fault tolerance due to redundancy. Redundancy ensures a task is completed even when others fail during mission execution. In some instances, especially critical missions, swarm members (decoys) strategically position themselves to act as a shield protecting critical robots from danger, consequently discounting the risk of failing a mission. For example, in military operations, decoys would protect self-destruction (Kamikaze) or reconnaissance drones from air defense systems. The

*Corresponding author. Email: semongt@biust.ac.bw

idea is to overload defense systems with less important drones to give way for drones carrying mission-critical payloads to finish the assignment.

At the center of Mobile Robots Collective Intelligence are control methods known as policies, control schemes or algorithms. Throughout this paper, these words would be used interchangeably depending on the context. Control methods acting in a centralized or decentralized manner gives robots the cognitive ability to cooperate and produce collective motion. Centralized control methods are designed to run in centralized locations such as servers. These methods are normally resource intensive and are susceptible to network performance limitations. Decentralized control methods, however, are not resource intensive as they are designed to run on the robot onboard computer, removing dependency on the external network infrastructure. Unlike centralized methods, decentralized methods offer redundancy and experience limited network latency. Control methods are crucial because they stipulates how each robot should behave under different circumstances, taking into account mission objectives.

Virágh et al. [7] suggested that, similar to nature swarm systems, collective motion in a system of multiple mobile robots is guaranteed by the safe, favorable and secure maneuvers of an individual robot. Developing multi-robot control methods that fulfill these requirements is an ongoing challenge in the science research community. Different multi-robot system control methods have been suggested, but to the best of our knowledge, none of them have been adopted for commercial use. Proposed control methods are either resource intensive, making it difficult to use in robots with limited computational strength, or are unable to address cooperation requirements simultaneously for them to be adopted in commercial applications. It is demonstrated by Moshayedi et al. [8] that traditional algorithms such as A* and Dijkstra, when utilized for path planning in multi-robot collective motion, struggle to handle frequent environmental changes, lack adaptability, become computationally expensive in large multi-robot systems or networks, are ineffective for multi-objective tasks, and fail to account for energy constraints.

In response to the limitations preventing the adoption of multi-robot systems in civil applications, as discussed in the previous paragraph, § recent developments in robotics focus more on moving beyond the theory and toward implementing practical implementation of control methods on physical robots. This trend is demonstrated in [9–11]. Moshayedi et al. [9] focused on control approaches that rely on minimal use of sensors for navigation. The study introduced the Extended Fusion 1 Method (EFM1), an algorithm that computes navigation dynamics using aggregated inputs from camera and infrared (IR) sensors. Continuing on this theme, Moshayedi et al. [10] demonstrated that multiple light control methods can be combined in resource constrained robots to address multiobjective problems. The study indicated this by integrating the Beetle Antennae Search (BAS) algorithm used to estimate the energy-optimal path A*, to calculate the shortest path, and Simultaneous Localization and Mapping (SLAM) for map generation. Similarly, Moshayedi et al. [11] investigated how the simulated robot environment matched real world robotic behavior. To archive this; GMapping, which is an algorithm based on SLAM, was used by AGVs (Autonomous Ground Vehicles) for path planning, mapping, and localization purposes. The study sought to bridge the gap between simulation and reality by generating 2D maps applicable to both environments, enabling AGVs to determine their position and plan the path accordingly. The findings indicate that the GMapping algorithm generally performs better in simulation than in physical platforms, highlighting the need to improve simulation models to better reflect real-world conditions.

In the same spirit, high fidelity simulators are being developed to foster control method testing. Examples of these simulators include RotorPy for aerial robots [12], V-Rep general purpose simulation framework [13], and Webots [14]. There is a need to define standards that specify how platforms (simulation and hardware) should be developed to issue congruence and smooth Sim-to-Real transfer. The lack of standardization across robotic platforms (in both hardware and simulation) presents a challenge to transfer developed control methods from a simulation to physical robots without additional configurations and tuning. Consequently, designers in pursuit of developing platform independent methods often choose to avoid incorporating robot hardware dynamics model in control methods. This is based on the assumption that robots are already equipped with low-level controllers that handle the physical dynamics. While this simplifies the control method development, it can limit their adoption in real world applications. Therefore, understanding the strengths and limitations of existing control methods is essential to effectively address this gap.

Other studies focus on the development of cost effective robot hardware that can be used for multi robot systems. The Swarm-bots project sponsored by the Future and Emerging Technologies of the European Commission focuses on developing mobile robots that can cooperate to transverse a challenging terrain [15, 16], as illustrated by Figure 1 . Bitcraze developed a Crazyflie quadcopter which is an open source platform used for educational and research purposes [17], which is demonstrated by Figure 2. The Kobot robot hardware was designed to meet the multi robot system

Figure 1. A Swarm-bots robots transgressing a challenging terrain. Robots are equipped with flexible grippers that enables them to connect with each other as demonstrated in the picture [15, 16]

Figure 2. A miniature Crazyflie quadrotor. Crazyflie is an open source created for educational and research purposes [17]

requirements, such as having the ability to identify other robots, deposit marks, and read marks in the environment without explicit programming [18].

Despite these advances, the strengths and limitations of the existing control algorithms for multiple robots are not yet fully understood. Therefore, this paper reviews existing control algorithms for coordinating the collective motion of multiple mobile robots, examining their strengths and weaknesses, with the aim of providing a comprehensive understanding that can guide the development of more reliable, platform independent, and real-world applicable control methods.

1.1. Scope and Contribution

This paper presents a review of different methods suggested by scholars to enable collective motion of mobile robots. Most of the reviewed works in this domain of knowledge classify and discuss these methods without providing their mathematical formulation, making it difficult for scholars from different disciplines to apply them [1, 19], or focus on a specific function of a collective, such as path planning [20, 21]. This paper takes a general approach by presenting commonly used methods and how they are applied in the field of collective motion of mobile robots. Moreover, it presents the mathematical foundation of these methods to make it easy for readers of different disciplines to understand and build on in their own research. In addition, this paper offers guidelines for choosing a particular control method over others, and suggests the optimal control method to be adopted to enable effective collective motion of mobile robots. Finally, this paper highlights research gaps that need to be addressed in the future.

To enhance the readability of this work, control methods are categorized into Nature Inspired Algorithms, Self Propelled Particles, and Learning based. See Figure 3. This classification is motivated by fundamental differences in the way each family of control methods behaves. Nature Inspired algorithms apply fixed optimization rules to enforce collective motion, and Self-Propelled Particle centric methods models emergent behavior using simple interaction rules without learning from experience. The Learning based, by contrast, develops and optimizes control policies through learning by interacting with the environment. It is worth noting that, although some hybrid variants blur these boundaries, the taxomony reflects the primary mechanism of each approach. Examples of these hybrid variants include Pheromone learning-enhanced neural ant colony optimization [22], and Large Language Model Enhanced Particle Swarm Optimization for Hyperparameter Tuning for Deep Learning Models [23].

The body of this paper is structured as follows; Section 1 introduces mobile robots collective motion and the important concepts related to the subject. Section 2 focuses on the fundamentals of collective motion while Section 3 discusses nature-inspired algorithms. It presents a brief introduction of each algorithm and reviews how they are used in mobile robots collective motion. Section 4 presents works that are based on Reynolds and Vicsék models. Section 5 gives a snapshot of common learning based methods used in mobile robots collective motion. The section reviews the literature on how Artificial Neural Networks(ANN) and Reinforcement Learning(RL) are used in mobile robot collective motion. Section 8 discusses the findings. Finally, Section 9 concludes and presents gaps in the research that should be addressed.

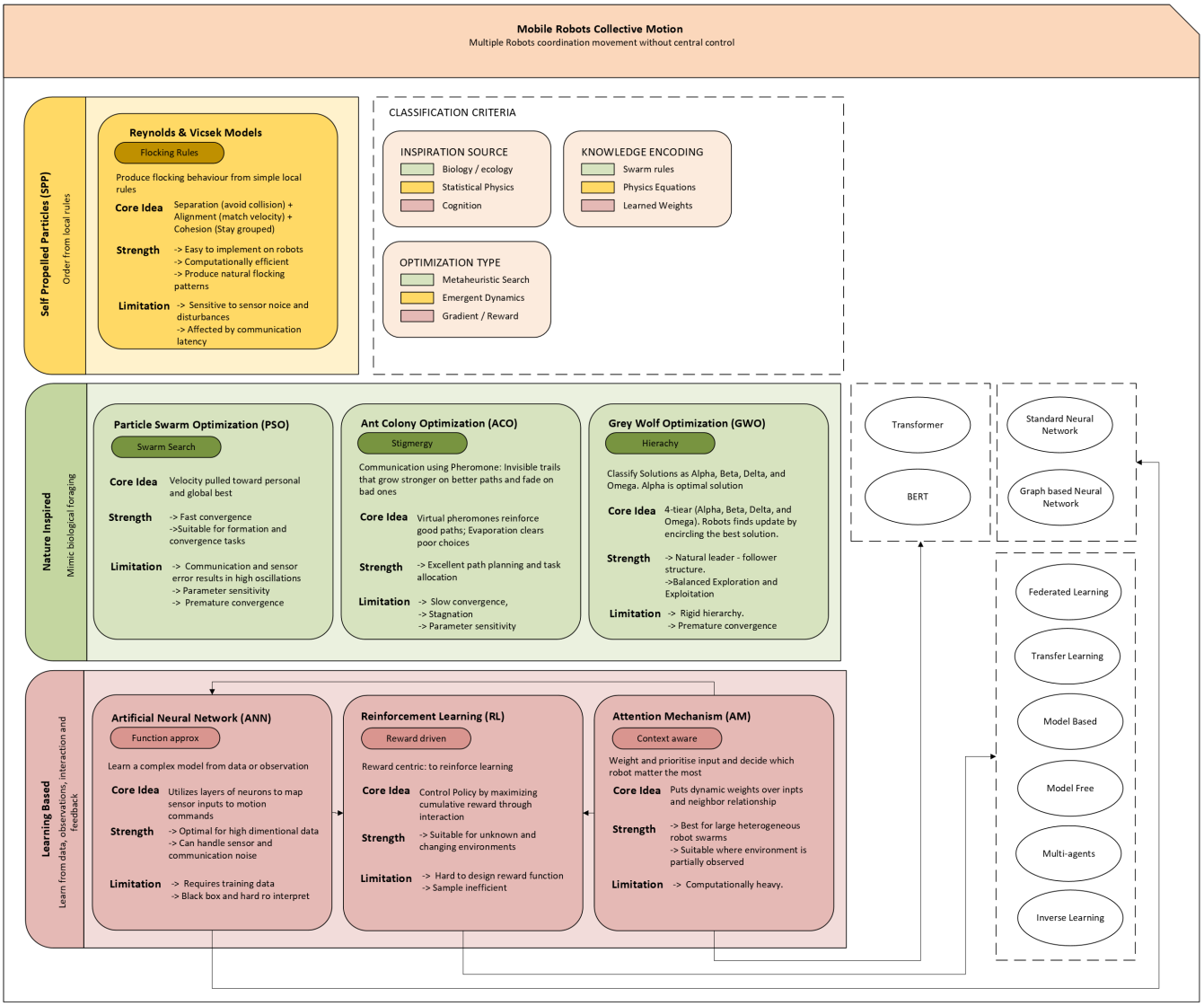

Figure 3. Represents three categories of control methods or algorithms. Categorization is based on fundamental differences in how each family of control methods behaves. Self Propelled Particles (SPP) algorithms model collective motion based on local flocking rules which are Separation, Cohesion, Separation, and sometimes Heading. Nature Inspired algorithms mimics the foraging behavior of organisms. Algorithms in Learning Based category learn an optimal strategy of producing collective motion from data, experience and observations

2. Fundamentals of Collective Motion

Collective motion alludes to the coordinated movement of multiple entities resulting in the display of complex patterns that are mesmerizing to watch. The phenomenon is widely experienced in natural systems, such as animal herds. Collective motion is not just about movement, for small creatures, it is also an important survival strategy. Gregarious creatures such as animals and insects rely on it as a defense strategy against predators and to hunt for large prey.

2.1. Biological Foundation

Formation vs Flocking based. In biological systems, collective motion can be expressed as flocking or cluster and formation based. Formation based systems maintain a well defined shape. As an example, migrating birds fly in V or linear formations. Formations have a functional advantage, such as aerodynamic efficiency [24]. V formation reduces drag force therefore helping migrating birds conserve energy when flying long distance. Again, formation based systems improve visual communication amongst members, resulting in improved navigation capability and group stability [25].

In contrast to formation based, flocking allows members to move together cohesively without maintaining a defined shape. Flocking exhibits complex emergent behavior by observing simple rules which are separation, cohesion, and alignment. Normally flocking based systems consist of autonomous members cooperating without central control system. Members make decisions based on the behavior pattern of their neighbors. Of-course, there are exceptions, e.g. bee swarms, where a leader is required to lead flocking entities.

Communication. Communication plays a crucial role in collective motion. The group members are using different methods to share knowledge. In some systems, members use chemical substances known as pheromone to communicate. Pheromones are deposited in the environment to signal danger or to mark trails. Others rely on visual signals like the waggle dances and light. These communication strategies encourage swarm members to act in a certain way. Poor communication can destabilize swarm system or even compromise its defense against predators.

Decision making. The success of swarm systems largely depends on their decision making ability, therefore, decision making is not a trivial issue. Group members consider tradeoff between physiological state (body condition, hormones), time, energy, environmental factors, and information from other members when deciding what to do next [26]. According to FengYing et al. [27], there are two common strategies that are generally used by natural swarm systems to reach an agreement. These strategies are centralized and decentralized consensus. The former considers the opinions of all swarm members where-else the latter, factors in the opinions of its close neighbors. Opinion can be expressed as a communication signal, for example, bee waggle dance and pheromone deposit, or behavioral dynamics like movement direction and speed.

2.2. Desired Properties in Artificial swarm systems

Natural swarms attributes are desirable in mobile robots collective motion for various reasons. These attributes are a result of years of evolution and adaptation. Therefore, mimicking them can aid development of control methods that are safe, reliable and robust for the envisioned artificial swarm applications. Natural swarm attributes comprise sharing knowledge, decision making, collective exploration especially for search and rescue missions, flocking and pattern formation, self-organization and emergent behavior, and parallelism. Parallelism enhances the speed and efficiency of solving a problem because it allows multiple agents to work on different parts of the problem simultaneously.

Mobile robots collective motion can either be classified as Unit Center Referenced, Leader Referenced, or Neighbor Referenced [6]. In leader referenced collective motion, robots within a swarm rely on a single robot that assumes the role of a leader to position themselves. Each robot in unit center referenced calculate a center of mass by taking into consideration the position coordinates of all robots participating in a swarm. This approach is not feasible for deprived resourced (computation, memory and storage) robots or in a large swarm system. In neighbor referenced, robots take into consideration the coordinates and the heading angle of a robot(single neighbor referenced) or robots (multi neighbor referenced) in its close proximity to calculate its position and the direction and disregard other members. This approach is resource efficient as compared to neighbor referenced.

3. Nature Inspired Algorithms

Nature has always been instrumental in inspiring the development of control schemes used for mobile robots collective motion. Scholars observed that the mesmerizing patterns displayed by natural swarms, such as bees and ants, are a result of each member acting on simple rules to be in sync with other group members. Motivated by this phenomenon, researchers have developed control methods known as Nature-Inspired Algorithms. In principle, these algorithms are meta-heuristic. They apply to distinct problems associated with optimization, task distribution, and coordination [28, 29]. According to Tang et al. [30], as demonstrated in Figure 4, these algorithms can be segregated into four categories, which are: Physics [31], Evolution, Biology, and Human Behavior based. This section discusses different nature-inspired algorithms used in artificial mobile robots collective motion.

3.1. PSO: Particle Swarm Optimization Algorithm

Particle Swarm Optimization (PSO) is among algorithms heavily utilized in mobile robots system collective motion. Equations 1 and 2 are a basic PSO as presented in [30, 32, 33].

$$\vec{v_i}(t+1) = w\vec{v_i}(t) + c_1 r_1(\vec{p_i} - \vec{x_i}) + c_2 r_2(\vec{g} - \vec{x_i}) \quad (1)$$

Equation 1 above calculates the velocity $\vec{v_i}$ of the ith particle in the next time step. To calculate velocity $\vec{v_i}$, Equation 1 add the inertia $w\vec{v_i}(t)$, particle personal influence $c_1 r_1(\vec{p_i} - \vec{x_i})$ and swarm or social influence $c_1 r_1(\vec{p_i} - \vec{x_i})$. The inertia term w is the influence measure of the previous velocity on the current velocity. c_1 and c_2 are random parameters with $[0, 1]$ range. r_1 weight the significance of the particle personal influence and r_2 to the social influence. Finally, $\vec{p_i}$ is the particle optimal position while $\vec{g}$ is the swarm optimal position.

$$\vec{x_i}(t+1) = \vec{x_i} + \vec{v_i}(t+1) \quad (2)$$

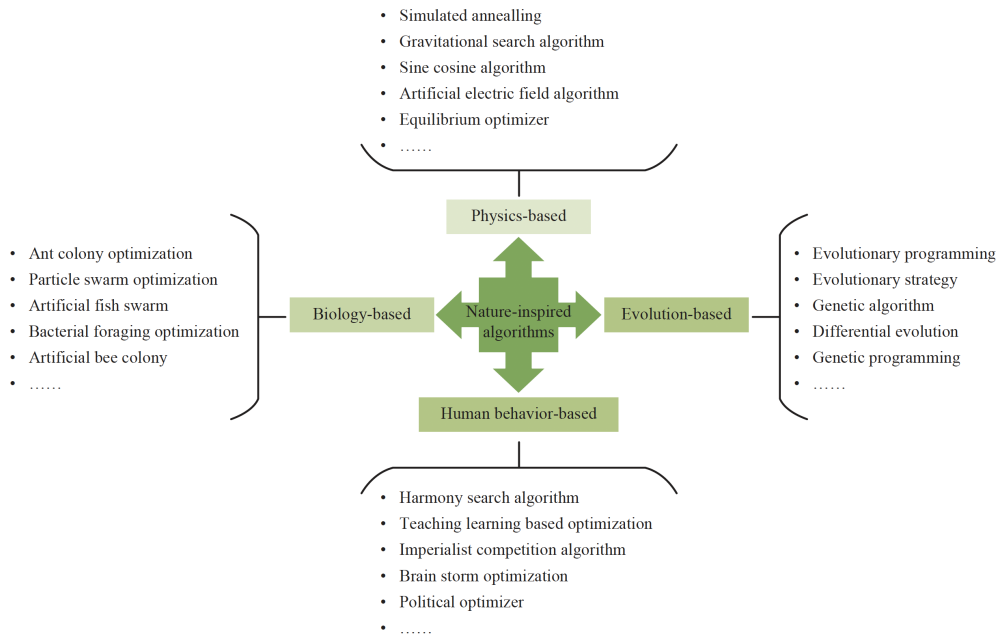

Figure 4. Shows four classes of Nat Algorithm and list examples of algorithms under each class [30]

To calculate the optimal position $x_i(t+1)$, the computed particle optimal velocity is added to the particle current position $\vec{x_i}$ as illustrated in Equation 2 above.

In multi-robot systems, PSO, combined with other algorithms, is used mainly for generating optimal paths. Wang Guoshi et al. [34] proposed PSO as a path planning scheme for multiple UAVs flying in threat invested environment. A new path is generated on the fly when a threat is encountered, otherwise, UAVs follow the static generated path. Zhang et al. [35] suggested an enhanced PSO variant called Fitness scaling Adaptive Chaotic Particle Swarm Optimization(FAC-PSO) for the Unmanned Combat Air Vehicles (UCAV) route computation. The generated path assist UCAVs avoid detection. Hybrid Particle Swarm Optimization and Genetic Algorithm [36] are adopted by Duan et al. [37] for generating UAV swarm formation. However, the scheme has a limitation with processing continuous input. To compensate for this, inputs are piecewise linearized. Ghamry et al. [38] adopted PSO integrated with parametrization and time discretization (PSO-CPTD) to compute the best flight trajectory. Rauch-Tung-Striebel(RTS) smoother, Metropolis Criterion, and PSO are merged by Wu et al. [39] to create the Robust Particle Swarm Optimization (RPSO) scheme. RPSO is introduced to compute a smooth and cost effective route for a fixed-wing UAV swarm flying in 3-dimensional space with static obstacles. In this case, PSO plans the path, Metropolis Criterion is used for enforcing convergence and alleviating the likelihood of being trapped in a local minima. RTS, on the other hand, reduces sharp curves in the generated path. In this work, while RPSO offers significant advantages such as enhanced path smoothness, the algorithm suffers from computational complexity as a result of integrating RTS and Metropolis Criterion with PSO.

To improve optimal path computation speed for Internet of Drones(IoD) flying in 3-dimensional spaces, just like in [28], Ahmed et al. [40] proposed utilizing chaos logic maps to initialize PSO, adaptive mutation to balance the exploitation and exploration efforts, and substituting active particles with new particles. The substitution process helps the PSO algorithm avoid giving suboptimal results and encourages it to compute global optimal solution. Manh et al. [41] introduced a motion encoding mechanism in PSO to track mobile targets. The algorithm converts the search space into motion space to optimize the search performance. However, increasing the number of swarm elements exponentially increases computational cost, making the algorithm unsuitable for use in large search spaces.

He et al. [42] developed a Hybrid Particle Swarm Optimization (HIPSO-MSOS) algorithm to compute low cost and smooth route for a group of UAVs flying in a tight 3-dimensional space. Improved Particle Swarm Optimization (IPSO) initialize particles representing the solution space, Symbiotic Organism Search (SOS) manage the interaction of particles by manipulating their position vectors, and a B-spline curve

smoothen sharp edges in the generated path. To simplify path coordination cost, time-stamp segmentation is employed to identify a common time reference in all UAVs. HIPSO-MSOS maintains standard PSO exploration strength with better convergence accuracy. Due to SOS algorithm, HIPSO-MSOS has poor convergence speed compared to the standard PSO. Mesquita and Gaspar [43] used PSO with Haversine formula [44] for path planning and path trajectory optimization based on the way points generated by the mission planner. The primary reason for using PSO and Haversine in optimizing UAV flight trajectories is to increase flight time by reducing mission cost. Despite this, algorithm performance decreases when more way points are added to the flight trajectory, and a larger way point radius increases the time to finish the mission. Lastly, the solution does not account for terrain dynamics and changing weather conditions.

Ming et al. [45] merged a Genetic Algorithm(GA) with PSO to resolve shortcomings relating to UAV swarm route computation and task allocation in the marine environment. GA introduces partial matching crossover and secondary transport mutation to optimize PSO velocity and position computation mitigating a chance of producing suboptimal results and to accommodate the uniqueness of the marine search space. PSO-based path planning algorithms, namely VAINDIWPSO and IC-VAINDIWPSO, were developed by Chu et al. [28] for UAV swarm path planning in a 3-dimensional threat invested space. For both algorithms, PSO was modified by introducing nonlinear dynamic inertia weights (INDIW). This approach improved the PSO fitness function and the convergence speed. Chaotic initialization and the logistic chaotic map are used in IC-VAINDIWPSO to generate a smooth path. Yu et al. [46] adopted Stochastic Dynamic Particle Swarm Optimization (SDPSO), which is a hybrid of the PSO algorithm and Simulated Annealing [47], to compute cost effective path for multiple UAVs flying in a space with many threats. To prevent producing suboptimal solution, Simulated Annealing (SA) computes best possible best solution. However, SA scheme introduces unnecessary particle oscillation, resulting in slow convergence. To reduce particle oscillation, Dimensional Learning Strategy(DLS) is introduced in SDPSO.

3.2. ACO: Ant Colony Optimization Algorithm

Ant Colony Optimization algorithm is a population based algorithm that mimics the foraging characteristics of natural ant colonies [48]. In order to reduce computational cost, ACO approximate an optimal solution, mainly, to solve a Travel Sales-man Problem (TSP). TSP challenge seeks to find an optimal way to visit points in a search space [49]. Similar to Particle Swarm Optimization(PSO), ACO is normally applied to solve optimization and search problems. Though different variants of ACO has been proposed for different problems in the literature, Equations 3 and 5, as presented by Gaertner et al. [50], are used in this section to briefly explain how ACO works before investigating how it is utilized in collective motion of mobile robots. More information on ACO can be found in [48, 51].

$$P_{ij}(t) = \begin{cases} \frac{\left[\tau_{ij}(t)\right]^{\alpha} * \left[\eta_{ij}\right]^{\beta}}{\sum_{k \in alw_k} \left[\tau_{ik}(t)\right]^{\alpha} * \left[\eta_{ik}\right]^{\beta}} & \text{if} \in alw_k \\ 0 & otherwise \end{cases} . \quad (3)$$

where

$$\eta_{ij} = \frac{1}{distance(i,j)} \quad (4)$$

Given multiple path that can be taken from point i to point j, Equation 3 calculate the probability $P_{ij}(t)$ of choosing a certain path over others. The probability is computed taking into consideration the amount of pheromone (trail) $\left[\tau_{ij}(t)\right]^{\alpha}$ deposited and the inverse distance $\left[\eta_{ij}\right]^{\beta}$ between the two points(i and j) computed by Equation 4. Using ACO, an artificial agent or a robot is likely to take a path with high pheromone level and a short distance. α and β parameters controls the influence of the pheromone and the distance. $\sum_{k \in alw_k}$ are all path allowed to be taken by agent k to point j;

$$\tau_{ij}(new) = \rho * \tau_{ij}(old) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (5)$$

where

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k(t)} & \text{if} \in alw_k \\ 0 & otherwise \end{cases} . \quad (6)$$

In order to avoid giving suboptimal results or getting stuck in a local minima, deposited pheromone evaporate over time. As a result, equation 5 is utilized to update pheromone. ρ is a evaporation parameter preventing infinite accumulation of pheromone, $\tau_{ij}(old)$ is an old pheromone value, m represent a number of ants in a system, and $\sum_{k=1}^{m} \Delta\tau_{ij}^{k}$ is the amount of pheromones deposited by ant k. $\Delta\tau_{ij}^{k}$ is calculated using equation 6. Q is a constant value and L_k is point i and j difference for all path allowed to be traveled by ant k.

Ant Colony Optimization Algorithm (ACO) is endorsed as a control scheme for artificial swarm systems, mainly to perform area search and path planning. It is worth mentioning that, despite the advantage of evaporating pheromones, in some cases, ACO has problem of early convergence resulting

in sub-optimal results. This problem surfaces when randomness is not introduced to balance exploration and exploitation. As a result, ACO will choose a greedy approach where a path with a high concentration of pheromone is favored.

Considering the aforementioned limitations of ACO, V-ACO which is an improvement of ACO is presented by Qiannan et al. [52]. V-ACO was designed to compute a path of formation based UAV swarm flying in an environment with known static threats. Cekmez et al. [53] proposed multiple colony as an answer to generate 3D path for multi UAVs in an environment with multiple obstacle. Each colony maintains its own pheromone table. As a result, it is highly unlikely for the algorithm to get stack in local minima.

ACO is slightly modified by Yang et al. [54] to allow parallel exploration of the unknown environment looking for a target. The algorithm start by dividing the environment into way points. At the beginning of the search, each way point is initialized with a certain amount of pheromone deposit. Multiple UAVs used as exploration nodes keep and maintain a pheromone map which is updated as the UAVs are interacting with the environment. This is essential for UAVs not to visit way points visited by other UAVs.

Chaotic Ant Colony Optimization to Coverage (CACOC), a scheme based on ACO and Chaotic System, is proposed by Rosalie et al. [55] to control a UAVs searching a threat invested space. The aim of developing the algorithm was to expand the UAVs search coverage. Chaotic Dynamical System based on Rössler system [56] and logistic map replaced the randomness of ACO to produce deterministic yet unpredictable movements. This approach is important to generate movement patterns that can be predicted in the ground control station (GCS) for easy management but complicated to be predicted by enemies. CACOC allows UAVs to maintain pheromone maps to determine movement through way points. Rössler system return map is used when there is no pheromone deposit to help UAV determine the way point to fly to. This algorithm does not take in consideration the physical body dynamics and does not include the collision avoidance mechanism. UAVs avoid collision with each other by flying at different altitude. Rosalie et al. [57] extended CACOC algorithm by including UAV physical body dynamics to determine UAVs movements in a search space in order to improve exploration and coverage. CACOC algorithm, just like Rosalie et al. [55], continued to manage UAV fleet movement while Model Predictive Control (MPC) [58] was introduced to calculate the position of each UAV within a swarm.

Dentler et al. [59] adopted CACOC as implemented by Rosalie et al. [55], added the UAV physical body dynamics as suggested by Rosalie et al. [57], and also added Collision Avoidance (CA) capability to build the CACOC+CA algorithm. Similar to Rosalie et al. [55], UAVs using CACOC+CA can navigate obstacles in the environment and avoid colliding among themselves, while using uniform altitude. Model Predictive Control [60] with Collision Avoidance (MPC-CA) was used to estimate UAVs position as well as to prevent UAVs from colliding with dynamic obstacles. CA mechanism employed in the study utilizes a potential field approach, causing the effectiveness of CACOC-CA to degrade as more members are added to the swarm.

CACOC algorithm [61] was used to coordinate the movement of UAVs within a swarm while Attractor Based Inter-Swarm collaborationS (ABISS) allow the UAV swarm to cooperate with Unmanned Ground Vehicle (UGV) swarm. An attractor is deposited when a swarm seeks assistance from other swarms to explore areas it cannot explore. Based on this condition, other swarms may choose to accept or reject the request. To further enhance CACOC, Rosalie et al. [62] introduced Bayesian for parameters optimization of the Rössler system fused in CACOC algorithm. Rössler system parameters are manipulated to improve the chaotic behavior of the CACOC as a result positively impacting the movement of UAVs in a search space.

Two hybrid ACO based algorithms named Ant Colony System Variable Neighborhood Descent(ACS-VND) and Max-Min Ant System Variable Neighborhood Descent(MMAS-VND) were employed by Kyriakakis et al. [63] to address Cumulative Capacitated Vehicle Routing Problem(CCVRP) [64] in multi-vehicle system. VND algorithm [65] appeared in both ACS-VND and MMAS-VND, and improved the solutions generated by ACO. Combination of ACS-VND and MMAS-VND in solving CCVRP improve quality of the solution, result in faster convergence and offers improved performance when compared to Ant Colony Optimization (ACO), Max-Min Ant System (MMAS), Variable Neighborhood Descent (VND) and Variable Neighborhood Search(VNS).

Another ACO variant named Improved Heuristic Mechanism ACO (IHMACO) algorithm in [66] computed the optimal path for different types of mobile robots. Addressing ACO limitations, IHMACO introduced; 1) adaptive pheromone concentration setting for coordinating units in the initial searching state, 2) directional judgment heuristic mechanism for improving searching and for reducing sharp edges, 3) better pseudo-random transfer strategy for improving convergence speed, and 4) dynamic adjustment of the pheromone evaporation rate to reduce sub-optimal results. IHMACO cost function considers two parameters which are the path length and turn times.

3.3. GWO: Grey Wolf Optimizer Algorithm

Grey Wolf Optimizer (GWO) was introduced and explained by Mirjalili et al. [29] as an population based optimization algorithm that mimics the social and hunting behavior of grey wolves pack. Grey wolves strictly adhere to social hierarchy consisting of the alpha α, beta β, delta δ, and omega ω levels. Alphas are responsible for making decisions and maintaining order. Beta wolves assist alphas and stand a chance to compete to become an alpha when one of the alphas dies. Delta wolves submit under alphas and betas but rank above omegas. Normally deltas are responsible for hunting, scouting and defending the pack. Omega wolves are the lowest ranking responsible for maintaining harmony by satisfying the dominance needs of alpha, beta, and delta wolves. Different versions of GWO have been introduced ever since 2023.

Equation 7 and 8 models the hunting behavior of GWO algorithm. The word hunting refers to an act of searching for the optimal solution.

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \tag{7}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{8}$$

$\vec{X}_p(t)$ represent prey or an optimal solution position in the search space where-else $\vec{X}(t)$ is the position of a grey wolf at time t. Since the optimal solution or prey position is unknown, GWO dim alpha α position as the optimal solution. This elevate the influence of alphas when searching for an optimal solution. $\vec{A}$ and $\vec{C}$ are coefficients computed by the Equation 9 and 10 respectively.

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{9}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{10}$$

Random vector $\vec{r}_1$ and $\vec{r}_2$ are in [0,1]. $\vec{r}_1$ and $\vec{r}_2$ make GWO stochastic. Minimizing $\vec{A}$, GWO adopt greedy behavior by transiting from exploration to exploitation. $\vec{A}$ is minimized by linearly reducing $\vec{a}$ from 2 to 0. If $|\vec{A}| < 1$, GWO exploit the optimal solution and if $|\vec{A}| > 1$ GWO explore the search space. To avoid GWO converging into suboptimal solution, $\vec{C}$ containing random values in [0,2] introduces randomness to distance between the wolf and the prey or optimal solution in Equation 7 through out the search.

Based on the GWO ranking, the optimal solution is α, second best is β, third best solution is δ, and the poor solution is ω. As a result, based on Equation 7 and 8, solutions for the three levels (α, β and δ) are calculated as shown by equations 11 and 12 respectively.

$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|, \vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|, \vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \tag{11}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot (\vec{D}_\alpha), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot (\vec{D}_\beta), \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot (\vec{D}_\delta) \tag{12}$$

Equation 13 average the position computed in 12 to compute global position of the optimal solution position at time $t+1$.

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{13}$$

Task allocation among UAV swarm members using GWO is studied in [67]. In this work, Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) [68] ideas are fused in GWO to form PSO-GA-DWPA (discrete wolf pack algorithm with principles of particle swarm optimization and genetic algorithm). PSO and GA are meant to optimize GWO convergence spreed. For task allocation, PSO-GA-DWPA utilizes Integer Matrix Coding method. PSO-GA-DWPA has proven to be agile in finding optimal solution and have high precision than PSO and GWO algorithms. The performance become more prominent in high dimensional solution spaces. It is worth noting that PSO-GA-DWPA assumes search space information such as obstacles location is known in advance. As result, the algorithm is not applicable in mission scenarios where information about obstacles is not available.

Multi-Population Parallel Wolf Pack Algorithm (MPPWPA) [69] enhanced the PSO-GA-DWPA [67] by introducing sub-population and elite-population to further improve the ability to find optimal solution and to prevent the algorithm from producing suboptimal solutions. Elite-population sub-group contains the best performing candidates from the population. Subgroups capacitate MPPWPA to solve high dimensional discrete optimization problems. Each sub-group performs optimization independently. Based on these reasons, MPPWPA is proposed to solve task allocation challenges in a UAV swarm. Similar to [67], MPPWPA assumed targets were static. This limits the application of the algorithm to be in mission scenarios where dynamic targets are not scouted. Improved Grey Wolf Optimizer (IGWO) algorithm was presented in [70] to compute a path for multiple robots. In order to address GWO tendency of falling in a local optimum and slow convergence, this study focus on developing a better position update method to manage tradeoff between GWO exploration and exploitation. Adding on, adaptive weights are introduced to α, β and δ terms representing the social hierarchy structure in GWO.

A task allocation framework for multi-UAV system, based on GWO principle, was proposed by Ziheng et al. [71]. The framework address three objectives which are task assignment, path finding, and target

searching. Different algorithms are proposed for each sub-problem: Improved Hungarian Algorithm [72] for task assignment, Theta* Algorithm [73] for path finding, and Dynamic Artificial Potential Field Search (DAPS) Algorithm [74] for target searching. This framework proved that dynamic task allocation in a complex search space can be realized by adopting wolf pack hunting behavior in their natural habitat as a task allocation principle. However, due to different algorithms utilized, it is cumbersome to strike a tradeoff between the solution quality, computational efficiency, and real-time performance in dynamic task allocation scenarios.

A study by Obadina et al. [75], combined Grey Wolf Optimizer (GWO) and Whale Optimization Algorithm (WOP) [76] for offline tuning of parameters in Leader–Follower Robot (LFR) manipulator system with four degrees of freedom. The system is meant to synchronize the movement of the follower robot with its leader. Euler-Lagrange [77] formulation was used to develop equation of motion for LFR system while fuzzy PD + I tracks the position of the follower robot. Jing et al. [78] used GWO algorithm along with Immune Algorithm (IA) and Variable Neighborhood Search (VNS) [79] to address task parameter planning for electronic-optical(EO) UAV swarm performing search and rescue mission. This study took into account the characteristics of EO device to increase coverage. GWO is adopted to manage the tradeoff between global and local search. The optimum correlation of speed and altitude for maximum object detection was realized.

3.4. Summary

Nature-inspired or metaheuristics algorithms offer reliable, flexible, and computationally efficient tools for solving complex optimization problems that are challenging to address with traditional deterministic methods. These features make nature-inspired algorithms a good choice for addressing the objectives of mobile robots system collective motion. Literature presented in this chapter demonstrates that nature-inspired algorithms have been enhanced with other methods to address multi-robot collective motion objectives. This is because traditional metaheuristic algorithms, in this case PSO, ACO, and GWO, suffer from premature convergence, slow convergence speed, and falling into local minima resulting in suboptimal solutions. Suggested methods used to enhance traditional metaheuristic algorithms include using chaos particle initialization, parameter tuning automation, removing or replacing inactive particles with new particles to increase the prospects of finding optimal solution, and improving the position update process of particles.

4. Self-Propelled Particles (SPP) Algorithms

Over the years, Self-Propelled Particles (SPP) Algorithms have been used to simulate the collective behavior of natural swarm systems. A swarm agent (a cell or an animal) is modeled as a particle. Using SPP, Reynolds demonstrated that artificial swarms could display emergent flocking behavior similar to that of natural swarm systems such fish school by observing **Cohesion**, **Separation** and **Alignment** rules [80]. As illustrated by Figure 5, he demonstrated by running a simulation of multiple agents known as boids. However, his work did not take into consideration the stochasticity of the environment.

Improving on Reynolds' work, Vicksek introduced a perturbation term to simulate stochasticity in the collective motion of natural swarms [81]. The improved model is commonly known as Standard Vickek Model (SVM). Equation 14 shows how SVM compute velocity of agent $\boldsymbol{i}$ at time step $t+1$ relative to its neighbors' velocity within radius R.

$$\vec{v}_i(t+1) = v_0 \frac{\left\langle \vec{v}_j(t) \right\rangle_R}{\left| \left\langle \vec{v}_j(t) \right\rangle_R \right|} + perturbation \tag{14}$$

The term v_0 is an assumed constant velocity of swarm agents. $\langle ... \rangle_R$ represents the average velocity v_j of all agents within radius R. Rather than considering global information in calculating velocity, the agent utilizes local information generated by its nearest neighbors. $\frac{\left\langle \vec{v}_j(t) \right\rangle_R}{\left| \left\langle \vec{v}_j(t) \right\rangle_R \right|}$ returns a unit vector taking the angle of direction assumed by the majority of its neighboring members in a swarm. The agent i angle of direction is computed by Equation 15.

$$v_i(t+1) = v_i(t) + \Delta_i(t) \tag{15}$$

Where $\Delta_i(t)$ is a random number from the probability distribution interval $[-\eta\pi, \eta\pi]$ representing a level of perturbations($\eta < 1$) and $v_i(t)$ is an agent i angle of direction at time t calculated using Equation 16 below.

$$v_i(t) = arctan\left[\frac{\left\langle v_{j,x} \right\rangle_R}{\left\langle v_{j,y} \right\rangle_R}\right] \tag{16}$$

By so doing, the agent takes the averaged speed and angle of direction of its neighboring agents in the next time step. $v_{j,x}$ and $v_{j,y}$ are the agent i neighbor j velocity coordinate(x and y).

Another important term in SVM is the order parameter φ calculated by equation 17. Order parameter calculates how coherent agents are in a system. When φ is close to 0, it shows that particles move in a random direction. On the other hand, when φ is near 1, it

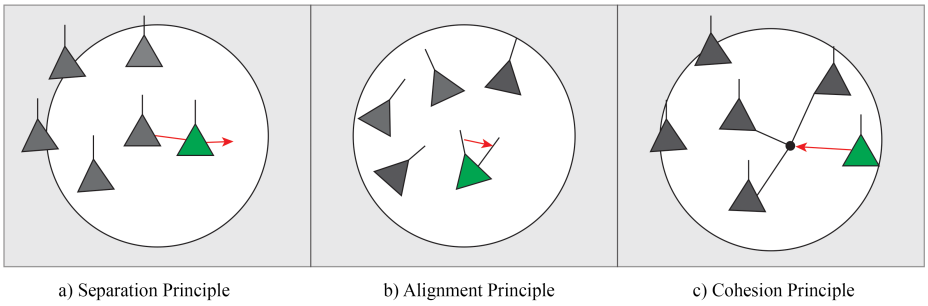

Figure 5. Principles shown in this figure mathematically models how an agent coordinates with it neighbors, within radius *r*, in a swarm. **Separation** helps agents to avoid collision, **Alignment** helps agents to match velocity, and **Cohesion** draws agents towards the center of the swarm

means most particles are moving coherently in the same direction.

$$\varphi \equiv \frac{1}{N v_0}\left[\sum_{n=1}^{N} \vec{v}_i\right] \tag{17}$$

By manipulating the SPP models (SVM), various models were proposed to model the behavior of different artificial swarm systems including mobile robots collective motion. Some models do not emphasize alignment rule and some explicitly model each rule separately [82]. This section discusses works that satisfies Self-Propelled Particles principles.

A decentralized model that satisfies the principles of flocking, introduced by Reynolds, was proposed by Chen et al. [83] to address drone swarm networks interconnection challenges. Drones acting as network nodes are either classified as a master (leader) or a slave (follower). Each master drone is associated to a single leader. Master nodes are responsible for determining the flight path, while slave drones determine an optimal position in relation to their master and close neighbors. Unlike in equation 14, where only one radius is used to determine the circumference where close neighbors should be positioned, three radii were used by Chen et al. [83]. Radius *r1* is the safe distance, radius *r2* ($r2 > r1$) is the optimal communication distance, and radius *r3* ($r3 > r2$) is the unstable communication distance of a drone *i* as demonstrated in Figure 6. The drone labeled *F* can calculate its velocity and position by considering the velocities of neighbors and its Master (Drone *L*) for it to stay in a shaded area.

The use of the pair-correlation or slave-master relationship adopted by Chen et al. [83], made it difficult to scale up a drone swarm. In other words, it encouraged the possibility of collision in a case where the drone count is scaled up. To address this problem, a multi-layer flocking control (SIMFC) [84] scheme was introduced to manage the topology of a drone network. SIMFC improved the flocking model proposed by Chen et al. [83] by introducing an additional layer to the drone network topology. The master and slave drones are separated into different layers, as shown in Figure 7

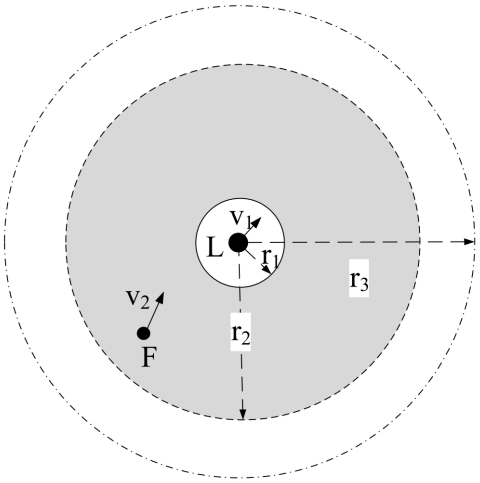

Figure 6. Shows the relationship of a slave drone *F* and its Master *L*

The dark grey, as shown in Figure 7, is an upper layer consisting of Master(Leader) drones and the root drone. The root drone determines the path of the overall swarm at a higher level. The lighter grey is a lower layer consisting of slave (follower) drones.

Sabine et al. [85], when investigating the impact of communication range on motion dynamics in artificial swarms, used a model built on the principles of SVM to guide the behavior of drones. In contrast to Chen et al. [83] and Dai et al. [84], the migration rule or principle was introduced to guide the drones to a destination point, as shown in Figure 8. The behavior of each drone, in a swarm, was influenced by the average direction of the neighboring drones and the migration principle.

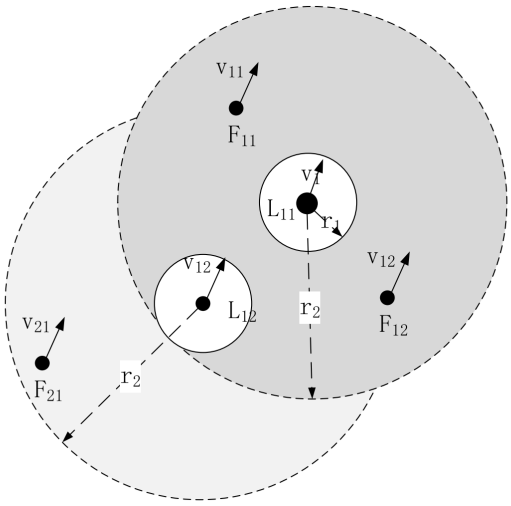

Figure 7. Is pictorial representation of SIMFC segregating master drones from slave drones. The upper layer(dark grey) consists of Master drones, and the lower layer(light grey) consists of Slave drones

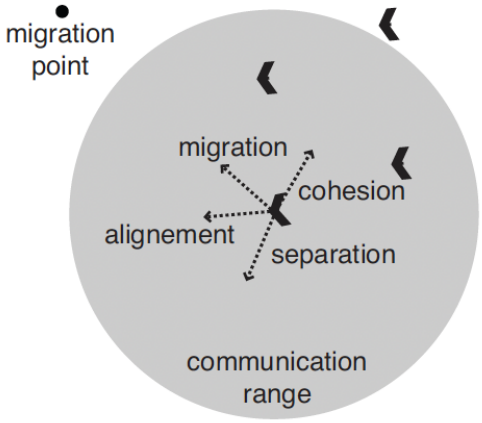

Figure 8. Demonstrate the influence of Alignment, Cohesion, Separation and Migration principles in a flocking behavior. A migration principle is introduced to help the drones to fly to the destination labeled as migration point. The grey circle around the communication range of the *ith* drone [85]

Similar to Hauert et al. [85], migration principle was adopted by Braga et al. [86] to help a group of drones fly t oward t he d estination o r m igration p oint. Like natural swarms such as birds, drones in Braga et al. [86] did not communicate their position and velocity. They perceive the environment around them using onboard sensors. The data from the environment, which includes the estimated position of the neighboring drones, was manipulated to help the drone determine its direction and velocity.

Another exceptional work in which actual drones were used was by Vasarhely et al. [87]. Drones using a $2\frac{1}{2}$ dimensional self-propelled algorithm developed in [87] performed the necessary calculations in an onboard computer and made necessary adjustments accordingly. In this case, $2\frac{1}{2}$ dimensional means that drones fly at different altitudes when landing and taking off. Gábor et al. [88] developed a configurable distributed SPP algorithm to control collective motion of an autonomous multi-UAV system in a confined environment. This study showed that the proposed algorithm was able to sustain stable collective motion. However, it was observed that parameter tuning was needed for desirable flocking behavior to be displayed when UAVs count was more than 30. Again, the algorithm developed in this study was not able to discount motion instabilities due to communication delays, wind, and inaccurate sensor readings.

Virágh, Csaba, et al. [7] introduced a two-dimensional solution based on Self-Propelled Flocking Algorithm and Target Tracking Algorithm to coordinate collective motion in a system comprising nine autonomous multi-rotor UAVs. Self-Propelled flocking algorithm facilitates the movement of units, whereas Target Tracking algorithm guides units toward a stationary target. The objective is to achieve an adaptive collective motion in an uncertain and noisy environment by considering real world factors such as communication errors and delays. Viscous friction term incorporated in these algorithms enhances velocity alignment among nearby drones in the swarm. Having an optimally tuned viscous friction (or friction-like) term helps damp velocity differences among units, enhancing swarm stability. Results demonstrated that the proposed solution resulted in stable collective motion in the simulated and physical environments. Experiments focused on wind, sensor noise, and communication delay. Communication failure and dynamic obstacles are not addressed.

4.1. Summary

Self-propelled particles (SPP) capture and explain the behavior of a system made up of multiple autonomous agents that move by self propulsion. In multi-robot collective motion, the SPP principle is applied to coordinate the movement dynamics in decentralized manner. The intention is to have multi-robot systems display emergent behaviors exhibited by natural systems. As demonstrated in the literature, SPP is suitable for decentralized systems consisting of robots having limited processing resources. This is a result of framework simplicity. Most of the control algorithms based on the framework are two-dimensional. In addition to cohesion, separation, and alignment rules, some studies have introduced

a migration rule. A migration rule helps a group of robots move towards a destination point. A virtual boundary, also known as a shill agent, is used to define the flying formation (square, circle, or ring). The drawback of the framework is that communication latency could introduce instability in SPP controlled systems. To address this, some studies suggest using the viscous friction term. The viscous friction term damps velocity differences among agents, thereby enhancing motion stability. Adjusting the viscous friction reduces oscillations, but does not completely remove oscillations. Incorporating more accurate feed-forward terms or learning capabilities can compensate for these communication delays and sensor errors proactively rather than solely relying on velocity damping. Better feedforward control enables the system to anticipate the required control actions ahead of time, effectively reducing the control relaxation time. A reasonable strategy for incorporating feed-forward terms and learning ability is to use Machine Learning (ML) to develop multi-robot control schemes or policies. Feed-forward terms can generalize across different environmental conditions when parameterized by ML.

5. Learning Based Methods

Artificial Intelligence (AI) class of algorithms mimics the cognitive reasoning of humans for optimizing decision making process [89, 90]. Through AI, artifacts are equipped to make decisions and to extrapolate future events by taking into account past and present experiences or by gaining insights from data [91]. Currently, AI principles are adopted in different domains such as Natural Language Processing (NLP), Image Processing(IM), Data Science, and Robotics. This Section, however, discusses literature where AI is utilized in mobile robots collective motion.

5.1. ANN: Artificial Neural Network

Artificial Neural Network Work (ANN) resembles behavior of the biological nervous system in solving complex problems [92]. ANN comprises bias, activation functions (e.g.,Sigmoid, Tanh, Relu, and Softmax) [93], and connected artificial neurons that behave like a biological neuron [94–96]. Activation functions transform signals exchanged between neurons in a network, and bias is introduced to the weighted sum to provide a means for each neuron to be adjusted. This process is important in making the network fit the training data, ensuring accurate predictions. During learning, training data traverse the network through a process called forward propagation. The loss function is applied to test how far(error margin) the output is from the expected answer. If error margin is high, parameters(weights and bias) of the ANN are adjusted

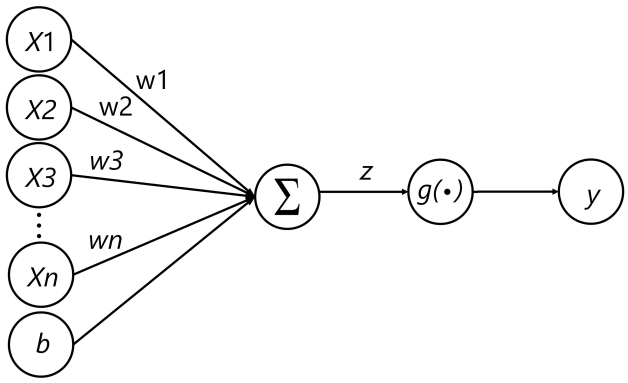

Figure 9. Single-layer perception showing how inputs are mapped into output. $g(\cdot)$ represents activation function

using gradient descent to improve the prediction. This process is called backpropagation. In some cases, ANN training can be implemented without backpropagation. For instance, in [97], Levenberg-Marquardt (LM) [98, 99] algorithm is proposed as an alternative to the traditional backpropagation training method.

Denoted by x, in Figure 9, are inputs to the neuron, w are weights, b is bias, z calculated by Equation 18, is the weighted sum of inputs. $g(\cdot)$ is the activation function.

$$z = \sum_{i=1}^{N} x_i w_i + b \tag{18}$$

Given the equation 18, y is calculated as $y = g(z)$.

Loss functions such as Mean Squared Error, Equation 19, and binary Cross-Entropy, Equation 20 are normally used to calculate an error or penalty when computation results are different from the expected answer.

$$mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(X_{\text{real}}(i) - X_{\text{predicted}}(i)\right)^2 \tag{19}$$

$X_{real}(i)$ represents an actual value, $X_{predicted}$ is a predicted value.

$$J_{bce} = -\frac{1}{M}\sum_{m=1}^{M}\left[y_m \times \log\left(h_\theta\left(x_m\right)\right) + \left(1 - y_m\right) \times \log\left(1 - h_\theta\left(x_m\right)\right)\right] \tag{20}$$

In the Equation 20, M is the number of training samples, y_m is the true label for the sample, h_θ is a neural network model parameterized by weights θ, and x_m input for training sample m.

There are variants of ANN such Graph Neural Networks(GNNs) [100] and Graph Convolutional Networks(GCNs) [101]. GNNs evolves from random walk models [102] and recursive neural networks [103–105] as a result, inheriting their attributes. On the other hand, GCNs are a type of GNN that extends

on Convolutional Neural Network [106] designed to work with graph-structured data [101, 107]. Compared to standard ANN, GNNs are suitable for complex unstructured data [108].

Modular Graph Neural Network Architecture (ModGNN) proposed by Kortevelesy et al. [109] builds on GNN concept to address flocking challenge in multi-agent systems. To achieve this, ModGNN aggregate data collected from neighboring agents to reduce redundancy, and allows user defined submodules to improve model generalization. Zhou et al. [110] introduced an extra controller using confidence score or ConfScore C_i, Equation 21, to improve the performance of the primary controller running non-learning and learning based policies.

$$C_i = \sum_{j \in N_i} \frac{v_i \cdot v_j}{\|v_i\| \cdot \|v_j\|} \tag{21}$$

Where N_i is neighbors of agent i,v_i is velocity vector of agent i , and v_j velocity vector of neighbors of agent i.

ConfScore C_i evaluates how good the position of an agent is and how best the velocity alignment of an agent is relative to its neighbors. High ConfScore is awarded to an agent with many neighbors and having velocity matching with the majority of its neighbors. In this study, ConfScore controllers enhance Graph Neural Network Controllers proposed by Tolstaya et al. [111] improved the flocking behavior of a multi-robot system. This approach is promising because agents remain organized even when they are moving at high velocity. However, communication range fluctuations disrupt cohesion and alignment.

Spaciotemporal Graph Neural Network (ST-GNN) proposed by Chen et al. [112] used Long Short-Term Memory (LSTM) [113, 114] to compute the behavior of each swarm member. The information passed in LSTM includes delayed states or historical information, current state, and information of its neighbors to compute swarm dynamics. ST-GNN can be applied to both centralized and distributed swarm settings. While the model performance is promising, using multiple delayed states ST-GNN increases the computational cost and memory.

A pheromone foraging algorithm based on a single layered neural network, referred to as Dynamic Wave Expansion Neural Network (DWENN) is developed in [115]. This algorithm model emergent behavior in a swarm of robots searching for a target based on a virtual pheromone communication principle [116–120]. Differential equations and DWENN capture system dynamics over time for parameter optimization purposes. This algorithm can be easily applied to other systems consisting of robots capable of making decisions based on local observations. However, this solution is not suitable for large swarms. Furthermore, robots using the algorithm struggle in dynamic environments and to maneuver around obstacles.

To address Multi-Robot system coverage control [121] problem, Agarwal et al. [122] proposed an Asynchronous Perception Action Control(APAC) framework. The framework consists of four modules: Perception, Inter-Robot Communication, Decentralized GNN message aggregation and inference, and a low-level controller. Perception module uses a Convolutional Neural Network (CNN) [123] to compute data from sensors for the GNN module. Inter-Robot Communication deals with information exchange between Robots, whereas GNN determines messages to be communicated. An aggregated GNN [124] model is used for this task. A low level controller made of multilayer perceptron (MLP) uses output from the GNN module to determine the action that robots are supposed to execute. Imitation Learning [125] using training data generated by the clairvoyant Lloyd algorithm [126] was used. Subsequently, APAC framework was compared with the centralized and decentralized Lloyd's algorithms [127]. The results proves that PAC is better than Lloyd's algorithms in performing coverage tasks. Furthermore, APAC outperforms Lloyd's algorithms in processing noisy position data, demonstrating robustness in dealing with uncertainties.

Akin to the APAC [122] framework is Learnable Perception-Action-Communication (LPAC) architecture proposed by Agarwal et al. [128]. The APAC and LPAC architectures vary in how GNN is implemented and their approach to execution timing. Distinct from APAC, LPAC combines inter-robot communication, and centralized GNN message aggregation and inference into what is known as the communication module. In other words, the communication module determines how robots communicate, message aggregation, and what to share. LPAC policy can be used in a new environment without degrading performance, can estimate robot positions in a noisy environment, and the framework scales well with large swarms. Nevertheless, LPAC exhibits poor performance when used in a small swarm size, is prone to performance degradation when the communication signal is inadequate, and LPAC, like LPAC-K3, with multi-hops in GNN layer can lead to increased computation and memory costs.

Uncontrolled oscillations and delayed consensus can occur in multi-robot collective motion due to communication breakdown or L-hop delayed states. Addressing this limitation, Chen et al. [129, 130] proposed ST-GNN (Spatio-Temporal Graph Neural Network). To achieve this, a unit considers its current state, information perceived from its immediate units, and the swarm historical information. ST-GNN can be used in varying swarm sizes due to flexible spatial and temporal expansion. However, the performance may be affected by limited information.

Liu et al. [131] addressed the navigation challenge in heavily crowded environments using a graph based neural network named the Decentralized Structural Recurrent Neural Network (DS-RNN). In this work, spatio-temporal graph (st-graph) was used to model crowd navigation to avoid the freezing robot problem experienced in robotic autonomous navigation systems based on Deep V-Learning baselines [132]. Although DS-RNN does not require a known or deterministic movement dynamics of the objects in the environment, accurately determining their position remain affected by sensor errors or noise.

In [133], Long et al. proposed CANet which is a collision avoidance policy for a system consisting of multiple robots interacting with an environment that has stationary objects and moving agents. CANet captures noisy sensor measurements to calculate the velocity and steering angle. The policy learn offline using training data generated by RVO2 simulator running the ORCA algorithm [134]. This study designed CANet to perform optimally using imperfect sensor reading. Since the policy is driven entirely by sensor measurements, intra and inter communication is not required for the policy to work. A core limitation of CANet is its reliance on training data for navigation behavior, which inevitably fails to capture all environmental dynamics and hinders real world deployment. Again, the utilization training data increases the chances of freezing robot problem where a robot considers all possible paths to be unsafe, as a result stop moving.

The Gumbel Social Transformer (GST) model introduced by Huang et al. [135] employed the Gumbel-Softmax sampling [136] to mitigate the freezing problem. GST predicts trajectories for robots that move through crowded environments. Through GST, Huang et al. aim to tackle two common limitations in existing trajectory prediction models. First, GST avoids the need to track every object in the environment by relying on partial observations of the environment. A Masked Long Short-Term Memory (LSTM) network together with graph-based structures compensates for missing information. Second, an Edge Gumbel Selector determines which objects in the environment should receive attention. Despite these advantages, GST still does not generalize well across different environments without retraining.

Deng et al. [137] proposed Adaptive Formation with Oscillation Reduction (AFOR), a method that allows multiple robots to adjust their formations while navigating complex environments. The policy is designed to reduce oscillations in the system as robots transition between formations. AFOR uses a two level hierarchical learning framework. At the upper level, a Graph Neural Network (GNN) encodes the spatial relationships among robots and propagates messages between them. At the lower level, a PPO-based controller trains individual robot agents to coordinate with others using the information provided by the GNN. Since the AFOR policy execute in a centralized manner, it inherits typical drawbacks of centralized architectures, such as a single point of failure. Moreover, the quality of the formation degrades as the number of robots increases.

5.2. RL: Reinforcement Learning

Reinforcement Learning (RL) is used in some studies to coordinate the collective behavior of artificial swarm artifacts. RL is a reward oriented method in which optimal policy producing the most rewarding actions is learned through trial and error [138–140]. This paradigm of machine learning is applicable to stochastic problems that can be modeled as a Markov Decision Process (MDP) [141]. MDPs are defined by tuple (S, A, P, R, γ) where S is set of states, A is set of actions, $P(s'|s, a)$ is transition probability, $R(s, a)$ is reward function, and $\gamma \in [0, 1]$ is the discount factor.

RL algorithms calculate the accumulated rewards, seek to optimize the policy by considering either quality of state (State-Value) or action taken from a state(Action-Value), and the reward. Therefore, the following equations, which are Return, State-Value, Action-Value, and Policy functions, are considered the main building blocks of RL.

Equations 22 and 23 calculate the Return. Return denoted by G_t is a function of the total accumulated rewards an agent gained beginning from time step t.

$$G_t = R_{t+1} + R_{t+1} + R_{t+1} + \cdots + R_T \tag{22}$$

The term R_T in the equation 22 represents the end of a task for episodic tasks. Episodic tasks have a clearly defined beginning and the end.

$$G_t = R_{t+1} + \gamma R_{t+1} + \gamma^2 R_{t+1} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{23}$$

Not all tasks in RL are episodic or finite; some tasks are infinite. To calculate G_t for such tasks, Equation 22 is modified to Equation 23 by adding a discount factor $\gamma \in [0, 1]$ and eliminating R_T representing the end of a task.

A Value Function $v_\pi(s)$ represented by Equation 24 evaluates the quality of the state by calculating the return G_t expected by an agent following policy π [142].

$$\begin{aligned} v_\pi(s) &= E_\pi[G_t \mid S_t = s] \\ &= E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \quad \text{for all } s \in \mathcal{S} \end{aligned} \tag{24}$$

Similarly, Action-Value Function $q_\pi(s,a)$ represented by equation 25 evaluates quality of action a taken by an agent in state s following policy π [142].

$$\begin{aligned} q_\pi(s,a) &= E_\pi[G_t \quad | \quad S_t = s, A_t = a] \\ &= E_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad | \quad S_t = s, A_t = a\right] \\ &\text{for all } s \in \mathcal{S} \quad \text{and for all } a \in \mathcal{A} \end{aligned} \tag{25}$$

Identifying the best policy π that maximizes expected returns for a given task constitutes the central objective of Reinforcement Learning (RL). Optimal Policy π_* is a policy that accumulates more rewards when compared to others. As an example, for State-Value function, policy π is better than or equal to policy $\pi^{'}$ when $v_\pi(s) \geq v_{\pi'}(s)$. Optimal State -Value function v_* is represented as

$$v_*(s) = \max_\pi(s) v_\pi(s), \quad \text{for all s} \in \mathcal{S} \tag{26}$$

,

where-else Optimal Action-Value function is represented as

$$\begin{aligned} q_*(s,a) &= \max_\pi(s) q_\pi(s,a), \\ &\text{for all s} \in \mathcal{S} \quad \text{and for all a} \in \mathcal{A}(s) \end{aligned} \tag{27}$$

.

For computation efficiency, q_* can be expressed in terms of v_* as follows

$$q_*(s,a) = E[R_{t+1} + \gamma v_*(S_{t+1}) \quad | \quad S_t = s, A_t = a] \tag{28}$$

.

Adding on Value functions presented in equations 26 and 27, Policy Gradient Theorem, equation 29, provides another avenue for the most rewarding policy to be learned by the agent. This method assists learning an optimal policy by directly tuning policy parameters θ. Policy Gradient Theorem is a foundation for policy optimization algorithms such as Asynchronous Advantage Actor-Critic (A2c/A3C) [143], Proximal Policy Optimization(PPO) [144], and Trust Region Policy Optimization (TRPO) [145].

$$\nabla_\theta J(\theta) \propto \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} q_\pi(s,a) \nabla \pi(a|s,\theta) \tag{29}$$

$\mu(s)$ is a deterministic policy that can be represented by $\pi(s)$, $q_\pi(s,a)$ gives the value of taking an action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ following policy π, $\pi(a|s,)$ is a stochastic policy, and θ denotes parameters such as weight in an Artificial Neural Network.

RL solutions in collective motion can either be expressed as Model Free (MFRL) or Model Based (MBRL) [146, 147]. The distinguishing factor between the two is that, MFRL does not incorporate environment model in the agent, while MBRL requires partial or full representation of the environment to be incorporated in the agent [148–150].

Single-Agent and Multi-Agent Framework(MARL) are used in RL to guide the learning process [151]. Figures 10 and 11 represent Single-Agent and Multi-Agent frameworks. The choice between the single agent framework and the multi agent framework largely depends on the number of artifacts required to address a problem. Single Agent framework is an approach where a single agent learn through interaction with the environment. When applied to a multi-agent system, each agent does not cooperate with others in learning but independently learn an optimal policy [152]. Transfer Learning can be adopted to allow cooperative learning between agents using a single agent framework [153]. In this case, an agent uses the previously acquired knowledge of others to improve its policy. MARL allows cooperative learning of multiple agents [154]. There are cases where MARL has been applied to guide learning of a single robot. As an example, Multi-Agent Reinforcement Learning for Single Quadruped Robot Locomotion (MASQ) developed by Liu et al. [155] is based on MARL. MASQ enhances cooperation of robot legs, where each leg is treated as an individual agent.

Multi-Agent Reinforcement Learning (MARL) framework is similar but distinct from Federated Reinforcement Learning (FRL). In FRL, multiple artifacts collaborate in learning global policy without sharing data.

Hong et al. [158] utilized Twin-Delayed Deep Deterministic Policy Gradient(TD3) to perform energy efficient path computation for multiple UAVs in a large continuous space. In comparison to conventional trajectory planning schemes like Dijkstra algorithm [159, 160], the enhanced TD3 model performs better in terms of utilizing UAV energy and computing optimal energy path. Goh et al. [161] introduced Deep Recurrent Q-Network (DRQN) as a model designed to control multiple UAVs tracking a subject while maintaining formation in aerial photography. The model introduces recurrent layers to the Deep Q-Network principle for preserving previous experience. Venturini et al. [162] adopted a MARL framework and a Deep Q-Network(DQN) [163] to coordinate UAVs searching for static targets in a complex environment. The proposed solution decentralizes decision making and promotes collaborative learning.

Ataur et al. [164] proposed the Economic Q Learning Algorithm that generate optimal route for multiple UAVs. Q-Learning algorithm, executing on each UAV, is used for learning the policy, while economic trading theory optimizes route planning by enhancing collaboration and task delegation based on economic principles. Long et al. [165] proposed a collision avoidance policy that uses sensor measurements to estimate the positions and velocities of other robots,

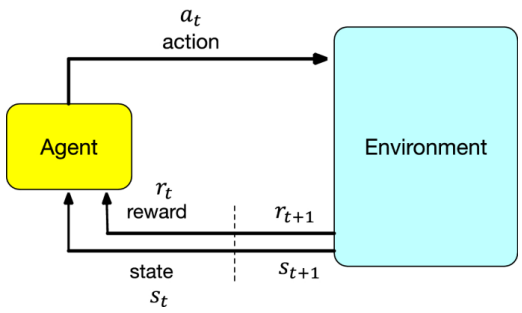

Figure 10. Single agent reinforcement learning architecture [156]

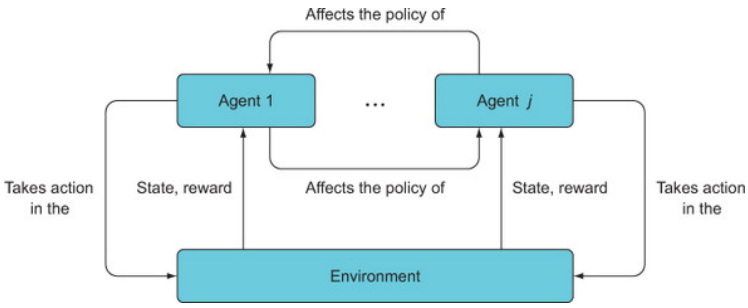

Figure 11. Multi agent reinforcement learning framework architecture [157]

as well as obstacles in the environment. The policy assumes a setting in which robots obtain position and velocity information through sensing rather than communication. Curriculum Learning (CL) is employed to train a Proximal Policy Optimization (PPO) agent. This study seeks to narrow the performance gap between centralized and decentralized multi-robot systems by adopting a parallel based approach to PPO implementation instead of CTDE. Although CTDE policies are usually classified as distributed, their reliance on centralized training makes them susceptible to the typical challenges associated with centralized architectures. Batra et al. [166] proposed a MARL [167, 168] policy based on the Proximal Policy Optimization(PPO) algorithm [144] to coordinate a swarm of UAVs displaying agile flying behavior in tight formations. Zhao et al. [169], utilized end-to-end RL [170] in reconnaissance mission planning for a pair of UAVs targeting high value mobile objects. In this context, Proximal Policy Optimization (PPO) is used for training the agent, and the clip function is used as a reward regularization technique to optimize the planning model.

Federated Reinforcement Learning (FedRL) [171, 172] was used Lee et al. [173] to facilitate distributed UAV swarm learning based on the PPO algorithm. In this study, FRL contributes by enhancing learning autonomy, where each UAV participates in collaborative learning without continuous data exchange. Abpeikar et al. [174] used RL to enhance Iterative-Transfer-Learning [175] for configuring collective motion in multi-robot systems. Iterative-transfer-learning based on Kullback-Leibler Divergence(KLD) [176] was applied on Collective Motion Tuning (CoMoT) [147] to optimize collective motion of a swarm of physical robots (Real Sphero BOLT) by learning from boids in a simulated environment.

Multi-Agent Deep Deterministic Policy Gradient(MADDPG) algorithm [177] and the Parallel Decoupling principle (PD-MADDPG) were used in [178] to control UAVs hunting enemy in combat grounds. PD-MADDPG algorithm is based on the Actor-Critic framework. Compared to MADDPG and the Independent Learning Deep Deterministic Policy Gradient(IL-DDPG), PD-MADDPG maximizes reward for the UAV swarm as a unit and for its individual members. This enhances collaborative efforts while improving individual UAV performance in a swarm. PD-MADDPG has improved convergence efficiency and exploration ability in continuous and dynamic environments. Yin et al. [179] developed Soft Actor-Critic [180] RL agent based on an off-policy strategy to control multiple mobile robots interacting with new terrains. Fuzzy logic controller is used in the algorithm to enhance obstacle avoidance ability, while curriculum prioritization optimizes learning by improving sampling efficiency.

5.3. AM: Attention Based Mechanism

Attention-based methods are optimization techniques that assign weights to different inputs or observations according to their relative significance. This enables an agent to concentrate on the most relevant input instead of handling all inputs equally, and forms the basis of architectures such as Transformers [181] and Bidirectional Encoder Representations from Transformers (BERT). Attention mechanisms (AM) are extensively used in deep learning and Multi-Agent RL (MARL). In MARL, they are mainly utilized to enhance agent communication and decision-making [182]. Integrating AM into resource constrained mobile robots can strengthen control policy performance by reducing power consumption and computational requirements, while simultaneously improving real time enhancing real-time decision-making in navigation, perception, and object search tasks [183].

Abdalwhab et al [184, 185], incorporated AM into the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm to enhance coordination in multi-robot systems. The attention-based encoder enables the critic network to aggregate and exploit information from all agents, yielding a more comprehensive representation of the multi-agent environment. Liu et al. [186], used

AM to make multi-robot exploration more efficient by avoiding redundant coverage of areas already visited by other robots and by selectively focusing on the robots that are most relevant to the current task. While Abdalwhab et al. [184, 185] encoded only current observations, [186] encoded observations o_i, actions a_i, and trajectories t_i. Escudie et al. [187], embedded a Graph Neural Network combined with AM within the policy for robot navigation. Jianliang [188] proposed the Res-ALBEF network, aimed at enhancing the educational capabilities of robots that act as student learning companions. Res-ALBEF integrates ALBEF (Align Before Use) and VGG19 with a dynamic Attention Mechanism: ALBEF aligns textual and visual embeddings to capture semantic relationships, VGG19 extracts visual features from images or videos, and the Attention Mechanism enables the model to concentrate on the input regions that are most relevant to the task.

5.4. Summary

Learning based methods present an opportunity to develop robust control schemes suitable for coordinating multiple robots. In the literature, most of the ANN control schemes are based on Graph Neural Network(GNN). While GNN such as LPAC-K3 offer improved performance, they suffer from high memory and computational requirements. This makes it challenging for GNN to be used in a system that has many members. Reinforcement Learning is another learning based method commonly used in the Muti-Robot control schemes. RL control schemes are suitable for navigation tasks where prior knowledge of the environment is not available. The MARL framework makes it possible for traditional RL algorithms such as Q-Learning to be used in multi-agent environments. Traditional RL algorithms are not designed to process large action-space. It is demonstrated, in studies where Deep Reinforcement Learning is used, that most existing DRL algorithms exhibit low sample efficiency when addressing large-scale or continuous state and action space problems.

6. Data Communication Constraints

Data communication Quality of Service (QoS) is crucial to the performance of multi-robot collective motion control algorithms. QoS explains how data arrive at the destination point, how much data is lost in transit, and how effectively data is delivered to the destination point [189]. These QoS elements have a direct influence on how control method behaves. For example, multi robot system using the ACO algorithm and its variants relies on a pheromone table updated collectively by all robots. Delay in data communication may result in fragmented or inconsistent pheromone map across robots, and poor convergence [190]. On the other hand, for PSO based systems, a robot relies on the velocity of other robots to calculate its social influence. Therefore, outdated data may results uncontrolled oscillations. Table 1 summarizes the impact of communication latency, Packet loss and Recovery ability of Nature Inspired, SPP, and Learning based methods in Multi-robot systems.

Table 1. Communication Sensitivity Comparison of GWO, PSO, and ACO in Multi-Robot Collective Motion Control

Factor	GWO	PSO	ACO	SPP	ANN	RL
Latency	High	High	High	High	Low	Low
Packet loss	High	High	Moderate	High	Low	Low
Recovery ability	Poor	Poor	Good	Moderate	Good	Good

■ High vulnerability ■ Moderate vulnerability ■ Low vulnerability

In most cases, QoS is determined by the chosen wireless networking technology and the communication architecture used. Table 4 in Appendix B summarizes the wireless standards that are commonly employed in robot networking. Since there is currently no wireless communication standard specifically designed for data exchange among mobile robots, the choice of networking technology is limited to conventional technologies such as Bluetooth and WiMAX. Therefore, given this limitation, it is crucial to assess how the choice of network technology and infrastructure affects the overall behavior of the robotic system and its energy consumption. From Table 1, it can be deduced that network infrastructures and technologies that rely on protocols such as UDP that maintain high packet loss are unsuitable for control strategies with limited recovery capabilities. Although UDP generally has lower latency than TCP in small-to medium sized robotic systems, its use in large scale systems can compromise stability [191]. In contrast, learning-based control methods can tolerate communication latency and high packet loss more effectively. These methods can learn to anticipate network latency as well as correct the errors that might occur as a result of packet loss.

7. Security Consideration

Beyond the communication latency issues discussed in Section 6, which threaten coordinated motion and cooperation in mobile robot swarms, the security vulnerabilities inherent in mobile swarm architectures present another risk that requires careful consideration. Neglecting to integrate security mechanisms into policy and communication architecture design can undermine collective motion and ultimately hinder the deployment of multi-robot systems in real-world applications. Multi-robot systems are prone or

susceptible to spoofing, deception, Byzantine failures, gradient poisoning in federated reinforcement learning, data injection, denial-of-service (DoS) attacks, and man-in-the-middle (MITM) attacks [192–194].

An approach to mitigate these threats is to reduce the attack surface by removing network-related risks. Instead of relying on policies that require data exchange over communication networks, one can adopt policies that employ signals or visual cues to coordinate robot behavior. Several works have employed these techniques to solve multi-robot coordination problems. For instance, Trujillo et al. [195] used image processing in a multi-robot system to localize all swarm members and track a leader, while Trujillo et al. [196] leveraged vision-based cues to coordinate multiple robots in GPS-denied environments. Similarly, Itani et al. [197] used acoustic signaling to coordinate robots with high spatial precision in 2D spaces. Although these methods are promising, they are not suitable for application scenarios that require information exchange between robots and communication with a ground station. In settings where both inter- and intra-swarm communications are necessary, security controls must be incorporated to protect the multi-robot system.

Effective countermeasures against cyberattack in multi-robot systems generally emphasize attack detection and adaptive defenses tailored to resource constrained robots. Detection techniques applicable to Deception and DoS attacks include residual based methods, machine-learning-based detectors, and observer based schemes. Defensive mechanisms include secure communication protocols that employ encryption to protect against spoofing and MITM attacks, switching control strategies [198] to counter deception and DoS, combinations of signature verification (e.g., the Elliptic Curve Digital Signature Algorithm (ECDSA)), consensus based optimization approaches [199, 200] to handle Byzantine failures. Among these controls, residual-detection-based schemes are well suited to systems composed of robots with limited computational resources, for example by using lightweight ciphers (e.g., PRESENT-128 with CMAC) to secure C2 payloads.

8. Discussion

Deciding on the best control scheme to use for coordinating collective motion presents a clear trade-off between computational overhead, robustness, and adaptability, see Table 2 and Table 3 in the Appendix A. There is no perfect method without limitations. Nature inspired are easier to implement, offer rapid convergence, but are often centralized and prone to giving suboptimal solutions. On the other hand, Self Propelled Particle (SPP) offer robust decentralized solution for emergent flocking behavior. However, SPP based control methods are not designed to handle sensor errors and communication delays. Sensor errors and communication delays result in high uncontrolled oscillations, causing the robot swarm system to be unsafe for real life application. While Learning-based methods can give optimal results in a delayed communication scenarios and presence of sensory error, they require high computational cost and lack of generalization often requiring retraining.

8.1. Algorithm Selection Guideline

The following guidelines should be considered when deciding on control scheme to employ for multi-robot collective motion:

- Employ properly enhanced Nature-Inspired Algorithms if the main goal of the control policy is path planning or task allocation. Techniques such as chaotic initialization can be adopted to improve convergence speed. Nature-Inspired Algorithms have been extensively studied, validated over time, and are generally reliable and straightforward to implement.

- SPP-based algorithms are well suited for generating emergent flocking behavior. However, these algorithms should only be applied when inter-robot communication is stable and dependable. The SPP algorithm has low computational overhead and produces naturalistic collective motion in multi-robot systems.

- For the robots that are operating in highly dynamic environments, where communication is intermittent or unreliable, learning-based approaches should be considered. Learning-based methods are typically resource intensive and techniques such as pruning, quantization, and distillation can be employed to compress the control policy so that it becomes suitable for resource-constrained robots. These methods reduce the size of the policy network by decreasing the number of parameters.

8.2. Barriers to Industrial and Commercial Adoption

Self-governing mobile robot swarms capable of coordinated movement experience intertwined technical, economic, regulatory, and socio-ethical obstacles that hinder civil, industrial, and commercial deployment. The reviewed literature indicates that only a small fraction of the proposed control policies have been validated on physical robots. Furthermore, many of the policies tested on physical platforms are tailored to specific missions. These deployment and sim-real-gaps undermines confidence in real world use and adoption.

Table 2. Comparison of Robot Control Schemes: Strengths and Weaknesses

Control Scheme	Strengths	Weaknesses	Robot Count
Nature Inspired Algorithms	Simple to implement, used for path planning, fast convergence, scalable, suitable for task allocation	Centralized variants, parameter sensitivity; can fall into local minima or give sub-optimal results.	**2 – 100.**
Self-Propelled Particles (SPP) Algorithms	Decentralized and distributed; easy to configure; produce naturalistic flocking and dynamic group formation	Oscillations may increase with swarm size; sensor noise and inaccurate readings can cause high swarm oscillation.	**4 – 11.**
Learning-Based Methods	Suitable for dynamic environments; can learn control policies; scalable; handles sensory errors and delays	Computationally intensive; may require retraining to adapt to new environments.	**2 – 100.**

Beyond technical issues, regulatory and legal frameworks also present significant challenges. Existing safety standards and certification procedures are not tailored to self-governing multi-robot systems. For instance, the Civil Aviation Authority of Botswana (CAAB) mandates that each drone must be paired with a pilot, which discourages the adoption of autonomous multi-robot operations.

Mitigation against these challenges require standardized process to guide the development of hardware and high-fidelity simulation platforms for multi-robot system. Regulatory bodies such as CAAB should evolve from a single-robot perspective and focus more on safe adoption criteria of multi-robot system laboratories. Lastly, multi-robot system cybersecurity concerns should be addressed.

8.3. From SIM to Reality

Multi-robot control policies must be rigorously evaluated against environmental parameters to guarantee safety and compliance with the system specifications. Policies can be tested on physical robots, or evaluated in simulation. Testing on real robots offers high fidelity and an inherent level of trust. However, this approach increases the risk of damage to the hardware, is expensive to implement, and does not scale well. Simulated environments, on the other hand, provide a fast and cost-effective means of testing policies in a controlled setting, while also reducing the risk of damaging physical robots. Simulations normally has a very low fidelity. It is difficult to replicate all real-world physical conditions in simulations. Therefore, using simulations can result in the development of brittle policies that are more likely to fail when deployed on real robots. To prevent this, some studies combine both real-world and simulated testing for the evaluation and optimization of control policies, as illustrated by Figure 12.

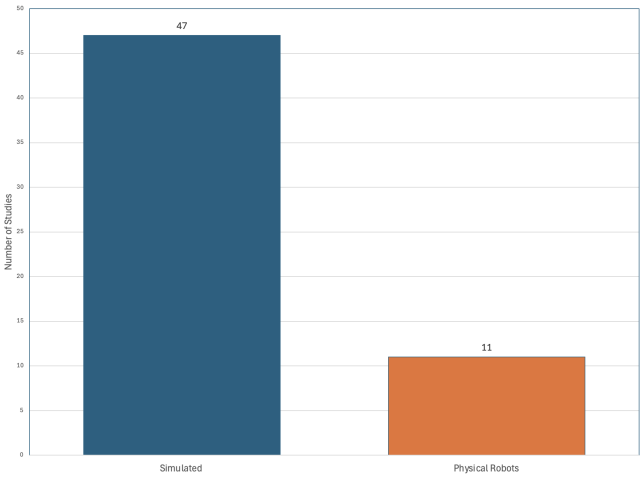

Figure 12. This captures the statistics of how simulation, and physical environments were used in the reviewed work

Figure 12 also shows a high reliance or usage of simulations compared to using physical robots. Out of 58 research papers, only 11 of them managed to transfer policies from simulation to physical robots, while 47 papers used simulations only. This indicate a SIM to reality gap that needs to be bridged.

To facilitate transfer from simulation to the real world, it is recommended to: 1) Apply Domain Randomization (DR) to emulate sensor noise. Progressively increase the degree of randomization and adjust parameters until a maximum acceptable threshold is reached. 2) Adopt TinyML. TinyML has been specifically designed for resource-constrained platforms such

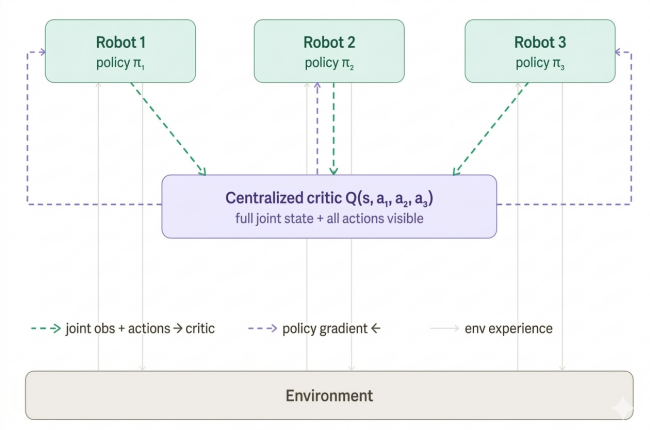

Figure 13. Shows CTDE framework. Policies are distributed across the robots. Critic algorithm, having full observation of the environment, train policies. This approach offload training responsibilities from the robots. Therefore making suitable to be used in resource deprived robots

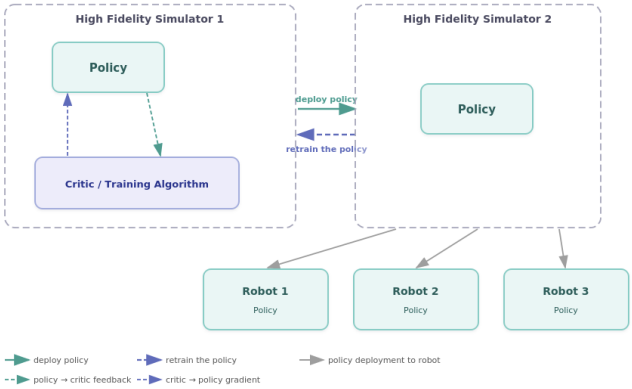

Figure 14. Shows the SIM to SIM framework. On this framework two high Fidelity Simulators, a policy is first trained, tested, and tuned in the first High Fidelity Simulator then deployed in the second High Fidelity for further testing. If the the policy doesn't perform as expected, it is returned back to the Simulator 1 for retraining

as Internet of Things (IoT) and edge devices. Salem et al. [201], applied TinyML to adaptive greenhouse lighting using a low-power microcontroller (ESP32). In the same spirit, Alongi et al. [202] introduced Deep but Tiny Neural Networks (DTNN) for atmospheric pressure forecasting on Tiny Embedded Systems. Furthermore, Cereda et al. [203] utilized TinyML to enable nano UAVs to perform vision-based human pose estimation. These studies demonstrated that the TinyML framework is well suited for systems composed of multiple resource-limited robots. 3) Employ high-fidelity simulators that can capture most real-world dynamics. 4) Refine the policy on physical robots after deployment. This phase of policy optimization is essential because simulations cannot fully represent all aspects of the physical environment. At this stage, depending on the policy development approach, fine-tuning may be performed manually or automated using training algorithm. For robots with limited computational resources, running both the policy and the training algorithm on-board for real-time parameter tuning is not ideal. Therefore, Central Training Distributed Execution (CTDE) [204] paradigm was adopted to offload le arning or pa rameter tuning to a centralized server. As demonstrated in Figure 13, the policies executed by individual robots are evaluated using centralist critic value function. The CTDE framework enabled continuous, real-time policy training, but it also created a single point of failure because the training algorithm was hosted in a central location.

Another method for bridging SIM to Reality gap involve deploying policies to physical robots without using online training algorithms. This eliminates a single point of failure associated with the CTDE framework. However, because this strategy limits continuous learning and policy adjustment, careful attention must be given to the design and training of the policy to ensure that it operates as intended. To achieve this, policies developed or learned in one simulated environment should be validated in a different simulated environment (sim-to-sim validation) before being deployed on physical robots. See Figure 14.

Digital Twin (DT) [205] technology presented another avenue for moving from simulation to the real world deployment experientially for Distributed Training Distributed Execution(DTDE). This approach allows a policy and learning to take place in the digital Robots while training data is received from a physical robots interacting with a physical environment. This arrangement can be utilized to perform mission with the policy receding in the digital asset. Alternatively, a learned policy can be transferred from digital twins and deployed to physical robots. This would eliminate communication overheads required between digital asset(robot) and its associated physical robot. See Figure 15.

9. Conclusions

This paper presented a review on control algorithms commonly used to coordinate collective behavior of multiple mobile robot. The objective was to review the strength and the limitation of these algorithms and to suggest what should be given attention when designing multi-robot control algorithms. From the literature, it is evident that there are two points to be considered when designing multi-robot control scheme. First is how robots are going to communicate. For robots to collaborate, they must inter-operate and be interconnected. Communication latency between

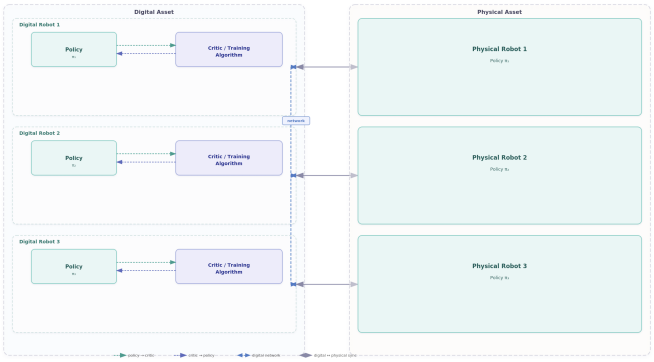

Figure 15. Shows a digital twin framework. According to the diagram, digital asset (digital robots) are tightly coupled with physical robots and the exchange information real time. Policy is trained in a digital asset. After training, a policy can be deployed in a physical robots or be used while it is residing in a digital asset

robots ultimately results in swarm instability. Since communication latency is unavoidable, to mitigate this challenge, it is important to develop control methods that have learning ability and feedforward terms. This way, swarm members will be equipped to learn the behavioral patterns of its neighbors making it possible to predict optimal action in the next time step. Second, algorithm architecture. Despite the fact that centralized control schemes remove computation responsibility form individual robots, they introduce single point of failure, and results in high communication traffic. Us ing decentralized or distributed algorithms encourage autonomy and reduces communication traffic be tween ro bots and control station. In distributed setting, each robot perform online real-time calculations by referencing a leader (leader referenced), using group center of mass (Unit center referenced), or by considering direction of motion and speed of its neighbors (Neighbor referenced). Neighbor referenced is resource efficient compared to Unit center referenced as a result suitable for deprived resourced robots. Unit center referenced computational requirements increase with the swarm size. Considering the limited processing capacity of robots, a computationally efficient al gorithm with low complexity is essential to mitigate the overall computational burden and to extend battery life.

References

[1] Navarro I, Matía F. A survey of collective movement of mobile robots. International Journal of Advanced Robotic Systems. 2013;10(1):73.

[2] Sasaki T, Biro D. Cumulative culture can emerge from collective intelligence in animal groups. Nature communications. 2017;8(1):15049.

[3] Cho J, Sung J, Yoon J, Lee H. Towards persistent surveillance and reconnaissance using a connected swarm of multiple UAVs. IEEE Access. 2020;8:157906-17.

[4] Wang Y, Bai P, Liang X, Wang W, Zhang J, Fu Q. Reconnaissance mission conducted by UAV swarms based on distributed PSO path planning algorithms. IEEE access. 2019;7:105086-99.

[5] Niculescu V, Polonelli T, Magno M, Benini L. Ultra-Lightweight Collaborative Mapping for Robot Swarms. arXiv preprint arXiv:240703136. 2024.

[6] Arkin RC, Balch T, et al. Cooperative multiagent robotic systems. Artificial intelligence and mobile robots. 1998:277-95.

[7] Virágh C, Vásárhelyi G, Tarcai N, Szörényi T, Somorjai G, Nepusz T, et al. Flocking algorithm for autonomous flying robots. Bioinspiration & biomimetics. 2014;9(2):025012.

[8] Moshayedi AJ, Roy AS, Khan ZH, Yang S, Razi A, Andani ME. Path Planning for AGVs: Balancing Computational Efficiency and Optimality. In: 2025 5th International Conference on Conference on Robotics, Automation and Intelligent Control (ICRAIC). IEEE; 2025. p. 1-12.

[9] Moshayedi AJ, Xu D, Sharifdoust M, Khan AS, Khan ZH, Andani ME. Comparative performance analysis of a novel fusion-based algorithm for AGV navigation. Advances in Computational Intelligence. 2025;5(4):1-31.

[10] Moshayedi AJ, Roy AS, Karan U, jun ZHANG M, Bassir D. An Integrated Beetle Antennae Search–Enabled Navigation Framework for Omnidirectional AGV Mobile Robots in Unknown Environments. EAI Endorsed Transactions on AI and Robotics. 2026;5.

[11] Moshayedi AJ, Li J, Khan AS, Khan ZH, Khoojine AS, Hu J, et al. Navigating the field: SLAM implementation for service robots in sports environment. In: Robotics and Artificial Intelligence in Sports Medicine and Sports Services. Elsevier; 2026. p. 241-75.

[12] Folk S, Paulos J, Kumar V. RotorPy: A Python-based Multirotor Simulator with Aerodynamics for Education and Research. arXiv preprint arXiv:230604485. 2023.

[13] Rohmer E, Singh SPN, Freese M. V-REP: A versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2013. p. 1321-6.

[14] Michel O. Cyberbotics ltd. webots™: professional mobile robot simulation. International Journal of Advanced Robotic Systems. 2004;1(1):5.

[15] Mondada F, Gambardella LM, Floreano D, Nolfi S, Deneuborg JL, Dorigo M. The cooperation of swarm-bots: Physical interactions in collective robotics. IEEE Robotics & Automation Magazine. 2005;12(2):21-8.

[16] Dorigo M. SWARM-BOT: An experiment in swarm robotics. In: Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005. IEEE; 2005. p. 192-200.

[17] Giernacki W, Skwierczyński M, Witwicki W, Wroński P, Kozierski P. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. In: 2017 22nd International Conference on Methods and Models in Automation and Robotics

(MMAR); 2017. p. 37-42.
[18] Turgut AE, Gokce F, Celikkanat H, Bayindir L, Sahin E. Kobot: A mobile robot designed specifically for swarm robotics research. Middle East Technical University, Ankara, Turkey, METU-CENG-TR Tech Rep. 2007;5(2007).
[19] Rubio F, Valero F, Llopis-Albert C. A review of mobile robots: Concepts, methods, theoretical framework, and applications. International Journal of Advanced Robotic Systems. 2019;16(2):1729881419839596.
[20] Sánchez-Ibáñez JR, Pérez-del Pulgar CJ, García-Cerezo A. Path planning for autonomous mobile robots: A review. Sensors. 2021;21(23):7898.
[21] Mohanty PK, Parhi DR. Controlling the motion of an autonomous mobile robot using various techniques: a review. Journal of Advance Mechanical Engineering. 2013;1(1):24-39.
[22] Wang H, Qin J. Pheromone learning-enhanced neural ant colony optimization. Neurocomputing. 2026:133375.
[23] Hameed S, Qolomany B, Belhaouari SB, Abdallah M, Qadir J, Al-Fuqaha A. Large Language Model Enhanced Particle Swarm Optimization for Hyperparameter Tuning for Deep Learning Models; 2025. Available from: https://arxiv.org/abs/2504.14126.
[24] Lissaman PB, Shollenberger CA. Formation flight of birds. Science. 1970;168(3934):1003-5.
[25] Heppner FH. Avian flight formations. Bird-banding. 1974;45(2):160-9.
[26] Harel R, Duriez O, Spiegel O, Fluhr J, Horvitz N, Getz WM, et al. Decision-making by a soaring bird: time, energy and risk considerations at different spatio-temporal scales. Philosophical Transactions of the Royal Society B: Biological Sciences. 2016;371(1704):20150397.
[27] FengYing Y, Din A, HuiChao L, Babar M, Ahmad S. Decentralized consensus in robotic swarm for collective collision and avoidance. IEEE Access. 2024;12:72143-54.
[28] Chu H, Yi J, Yang F. Chaos particle swarm optimization enhancement algorithm for UAV safe path planning. Applied Sciences. 2022;12(18):8977.
[29] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Advances in engineering software. 2014;69:46-61.
[30] Tang J, Liu G, Pan Q. A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. IEEE/CAA Journal of Automatica Sinica. 2021;8(10):1627-43.
[31] Can U, Alatas B. Physics Based Metaheuristic Algorithms for Global Optimization. American Journal of Information Science and Computer Engineering. 2015;1(3):94-106.
[32] Jiang Y, Hu T, Huang C, Wu X. An improved particle swarm optimization algorithm. Applied Mathematics and Computation. 2007;193(1):231-9.
[33] Bai Q. Analysis of particle swarm optimization algorithm. Computer and information science. 2010;3(1):180.
[34] Wang G, Li Q, Guo L. Multiple UAVs routes planning based on particle swarm optimization algorithm. In: 2010 2nd International Symposium on Information Engineering and Electronic Commerce. IEEE; 2010. p. 1-5.
[35] Zhang Y, Wu L, Wang S, et al. UCAV path planning by fitness-scaling adaptive chaotic particle swarm optimization. Mathematical Problems in Engineering. 2013;2013.
[36] Mathew TV. Genetic algorithm. Report submitted at IIT Bombay. 2012;53:18-9.
[37] Duan H, Luo Q, Shi Y, Ma G. ? Hybrid particle swarm optimization and genetic algorithm for multi-UAV formation reconfiguration. IEEE Computational intelligence magazine. 2013;8(3):16-27.
[38] Ghamry KA, Kamel MA, Zhang Y. Multiple UAVs in forest fire fighting mission using particle swarm optimization. In: 2017 International conference on unmanned aircraft systems (ICUAS). IEEE; 2017. p. 1404-9.
[39] Wu X, Bai W, Xie Y, Sun X, Deng C, Cui H. A hybrid algorithm of particle swarm optimization, metropolis criterion and RTS smoother for path planning of UAVs. Applied Soft Computing. 2018;73:735-47.
[40] Ahmed G, Sheltami T, Mahmoud A, Yasar A. IoD swarms collision avoidance via improved particle swarm optimization. Transportation Research Part A: Policy and Practice. 2020;142:260-78.
[41] Phung MD, Ha QP. Motion-encoded particle swarm optimization for moving target search using UAVs. Applied Soft Computing. 2020;97:106705.
[42] He W, Qi X, Liu L. A novel hybrid particle swarm optimization for multi-UAV cooperate path planning. Applied Intelligence. 2021;51:7350-64.
[43] Mesquita R, Gaspar PD. A novel path planning optimization algorithm based on particle swarm optimization for UAVs for bird monitoring and repelling. Processes. 2021;10(1):62.
[44] Prasetya DA, Nguyen PT, Faizullin R, Iswanto I, Armay EF. Resolving the shortest path problem using the haversine algorithm. Journal of critical reviews. 2020;7(1):62-4.
[45] Yan M, Yuan H, Xu J, Yu Y, Jin L. Task allocation and route planning of multiple UAVs in a marine environment based on an improved particle swarm optimization algorithm. EURASIP Journal on Advances in Signal Processing. 2021;2021:1-23.
[46] Yu Z, Si Z, Li X, Wang D, Song H. A novel hybrid particle swarm optimization algorithm for path planning of UAVs. IEEE Internet of Things Journal. 2022;9(22):22547-58.
[47] Rutenbar RA. Simulated annealing algorithms: An overview. IEEE Circuits and Devices magazine. 2002;5(1):19-26.
[48] Blum C. Ant colony optimization: Introduction and recent trends. Physics of Life reviews. 2005;2(4):353-73.
[49] Jünger M, Reinelt G, Rinaldi G. The traveling salesman problem. Handbooks in operations research and management science. 1995;7:225-330.
[50] Gaertner D, Clark KL, et al. On Optimal Parameters for Ant Colony Optimization Algorithms. In: IC-AI. Citeseer; 2005. p. 83-9.

[51] Dorigo M, Blum C. Ant colony optimization theory: A survey. Theoretical computer science. 2005;344(2-3):243-78.
[52] Qiannan Z, Ziyang Z, Chen G, Ruyi D. Path planning of UAVs formation based on improved ant colony optimization algorithm. In: Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference. IEEE; 2014. p. 1549-52.
[53] Cekmez U, Ozsiginan M, Sahingoz OK. Multi colony ant optimization for UAV path planning with obstacle avoidance. In: 2016 international conference on unmanned aircraft systems (ICUAS). IEEE; 2016. p. 47-52.
[54] Yang F, Ji X, Yang C, Li J, Li B. Cooperative search of UAV swarm based on improved ant colony algorithm in uncertain environment. In: 2017 IEEE International Conference on Unmanned Systems (ICUS). Ieee; 2017. p. 231-6.
[55] Rosalie M, Danoy G, Chaumette S, Bouvry P. From random process to chaotic behavior in swarms of UAVs. In: Proceedings of the 6th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications; 2016. p. 9-15.
[56] Gaspard P, et al. Rössler systems. Encyclopedia of nonlinear science. 2005;231:808-11.
[57] Rosalie M, Dentler JE, Danoy G, Bouvry P, Kannan S, Olivares-Mendez MA, et al. Area exploration with a swarm of UAVs combining deterministic chaotic ant colony mobility with position MPC. In: 2017 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE; 2017. p. 1392-7.
[58] Garcia CE, Prett DM, Morari M. Model predictive control: Theory and practice—A survey. Automatica. 1989;25(3):335-48.
[59] Dentler J, Rosalie M, Danoy G, Bouvry P, Kannan S, Olivares-Mendez MA, et al. Collision avoidance effects on the mobility of a UAV swarm using chaotic ant colony with model predictive control. Journal of Intelligent & Robotic Systems. 2019;93:227-43.
[60] Holkar KS, Waghmare LM. An overview of model predictive control. International Journal of control and automation. 2010;3(4):47-63.
[61] Stolfi DH, Brust MR, Danoy G, Bouvry P. Emerging inter-swarm collaboration for surveillance using pheromones and evolutionary techniques. Sensors. 2020;20(9):2566.
[62] Rosalie M, Kieffer E, Brust MR, Danoy G, Bouvry P. Bayesian optimisation to select Rössler system parameters used in Chaotic Ant Colony Optimisation for Coverage. Journal of computational science. 2020;41:101047.
[63] Kyriakakis NA, Marinaki M, Marinakis Y. A hybrid ant colony optimization-variable neighborhood descent approach for the cumulative capacitated vehicle routing problem. Computers & Operations Research. 2021;134:105397.
[64] Ke L, Feng Z. A two-phase metaheuristic for the cumulative capacitated vehicle routing problem. Computers & Operations Research. 2013;40(2):633-8.
[65] Duarte A, Mladenović N, Sánchez-Oro J, Todosijević R. Variable neighborhood descent. In: Handbook of heuristics. Springer; 2016. p. 1-27.
[66] Liu C, Wu L, Xiao W, Li G, Xu D, Guo J, et al. An improved heuristic mechanism ant colony optimization algorithm for solving path planning. Knowledge-Based Systems. 2023;271:110540.
[67] Lu Y, Ma Y, Wang J, Han L. Task assignment of UAV swarm based on wolf pack algorithm. Applied Sciences. 2020;10(23):8335.
[68] Lambora A, Gupta K, Chopra K. Genetic algorithm-A literature review. In: 2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon). IEEE; 2019. p. 380-4.
[69] Lu Y, Ma Y, Wang J. Multi-population parallel Wolf Pack algorithm for task assignment of UAV swarm. Applied Sciences. 2021;11(24):11996.
[70] Dong L, Yuan X, Yan B, Song Y, Xu Q, Yang X. An improved grey wolf optimization with multi-strategy ensemble for robot path planning. Sensors. 2022;22(18):6843.
[71] Wang Z, Zhang J. A task allocation algorithm for a swarm of unmanned aerial vehicles based on bionic wolf pack method. Knowledge-Based Systems. 2022;250:109072.
[72] Jonker R, Volgenant T. Improving the Hungarian assignment algorithm. Operations research letters. 1986;5(4):171-5.
[73] Firmansyah ER, Masruroh SU, Fahrianto F. Comparative analysis of a* and basic theta* algorithm in android-based pathfinding games. In: 2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M). IEEE; 2016. p. 275-80.
[74] Jayaweera HM, Hanoun S. A dynamic artificial potential field (D-APF) UAV path planning technique for following ground moving targets. IEEE access. 2020;8:192760-76.
[75] Obadina OO, Thaha MA, Mohamed Z, Shaheed MH. Grey-box modelling and fuzzy logic control of a Leader–Follower robot manipulator system: A hybrid Grey Wolf–Whale Optimisation approach. ISA transactions. 2022;129:572-93.
[76] Mirjalili S, Lewis A. The whale optimization algorithm. Advances in engineering software. 2016;95:51-67.
[77] Ortega R, Loria A, Nicklasson PJ, Sira-Ramirez H. Euler-Lagrange systems. In: Passivity-based Control of Euler-Lagrange Systems: Mechanical, Electrical and Electromechanical Applications. Springer; 1998. p. 15-37.
[78] Jing X, Hou M, Li W, Chen C, Feng Z, Wang M. Task Parameter Planning Algorithm for UAV Area Complete Coverage in EO Sector Scanning Mode. Aerospace. 2023;10(7):612.
[79] Mladenović N, Hansen P. Variable neighborhood search. Computers & operations research. 1997;24(11):1097-100.
[80] Reynolds CW. Flocks, herds and schools: A distributed behavioral model. In: Proceedings of the 14th annual conference on Computer graphics and interactive techniques; 1987. p. 25-34.
[81] Vicsek T, Zafeiris A. Collective motion. Physics reports. 2012;517(3-4):71-140.

[82] Strömbom D. Collective motion from local attraction. Journal of Theoretical Biology. 2011;283(1):145-51. Available from: https://www.sciencedirect.com/science/article/pii/S002251931100261X.
[83] Chen M, Dai F, Wang H, Lei L. DFM: A distributed flocking model for UAV swarm networks. IEEE Access. 2018;6:69141-50.
[84] Dai F, Chen M, Wei X, Wang H. Swarm intelligence-inspired autonomous flocking control in UAV networks. IEEE Access. 2019;7:61786-96.
[85] Hauert S, Leven S, Varga M, Ruini F, Cangelosi A, Zufferey JC, et al. Reynolds flocking in reality with fixed-wing robots: communication range vs. maximum turning rate. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE; 2011. p. 5015-20.
[86] Braga RG, da Silva RC, Ramos ACB, Reynolds CW. Development of a Swarming Algorithm Based on Reynolds Rules to control a group of multi-rotor UAVs using ROS; 2016. Available from: https://api.semanticscholar.org/CorpusID:221093766.
[87] Vásárhelyi G, Virágh C, Somorjai G, Tarcai N, Szörényi T, Nepusz T, et al. Outdoor flocking and formation flight with autonomous aerial robots. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE; 2014. p. 3866-73.
[88] Muiños-Landin S, Fischer A, Holubec V, Cichos F. Reinforcement learning with artificial microswimmers. Science Robotics. 2021;6(52):eabd9285.
[89] Hamet P, Tremblay J. Artificial intelligence in medicine. Metabolism. 2017;69:S36-40.
[90] Ertel W. Introduction to artificial intelligence. Springer; 2018.
[91] Ramesh A, Kambhampati C, Monson JR, Drew P. Artificial intelligence in medicine. Annals of the Royal College of Surgeons of England. 2004;86(5):334.
[92] Zhang Z, Zhang Z. Artificial neural network. Multivariate time series analysis in climate and environmental research. 2018:1-35.
[93] Sharma S, Sharma S, Athaiya A. Activation functions in neural networks. Towards Data Sci. 2017;6(12):310-6.
[94] Shanmuganathan S. Artificial neural network modelling: An introduction. Springer; 2016.
[95] Nwadiugwu MC. Neural networks, artificial intelligence and the computational brain. arXiv preprint arXiv:210108635. 2020.
[96] Shao F, Shen Z. How can artificial neural networks approximate the brain? Frontiers in psychology. 2023;13:970214.
[97] Wilamowski BM, Yu H. Neural network learning without backpropagation. IEEE Transactions on Neural Networks. 2010;21(11):1793-803.
[98] Ranganathan A. The levenberg-marquardt algorithm. Tutoral on LM algorithm. 2004;11(1):101-10.
[99] Moré JJ. The Levenberg-Marquardt algorithm: implementation and theory. In: Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977. Springer; 2006. p. 105-16.
[100] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE transactions on neural networks. 2008;20(1):61-80.
[101] Wu F, Souza A, Zhang T, Fifty C, Yu T, Weinberger K. Simplifying graph convolutional networks. In: International conference on machine learning. Pmlr; 2019. p. 6861-71.
[102] Weiss GH. Random walks and their applications: Widely used as mathematical models, random walks play an important role in several areas of physics, chemistry, and biology. American Scientist. 1983;71(1):65-71.
[103] Chinea A. Understanding the principles of recursive neural networks: A generative approach to tackle model complexity. In: International Conference on Artificial Neural Networks. Springer; 2009. p. 952-63.
[104] Goller C, Kuchler A. Learning task-dependent distributed representations by backpropagation through structure. In: Proceedings of international conference on neural networks (ICNN'96). vol. 1. IEEE; 1996. p. 347-52.
[105] Frasconi P, Gori M, Sperduti A. A general framework for adaptive processing of data structures. IEEE transactions on Neural Networks. 1998;9(5):768-86.
[106] O'shea K, Nash R. An introduction to convolutional neural networks. arXiv preprint arXiv:151108458. 2015.
[107] Chen M, Wei Z, Huang Z, Ding B, Li Y. Simple and deep graph convolutional networks. In: International conference on machine learning. PMLR; 2020. p. 1725-35.
[108] Scarselli F, Gori M, Tsoi A, Hagenbuchner M, Monfardini G. The Graph Neural Network Model. IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council. 2009 01;20:61-80.
[109] Kortvelesy R, Prorok A. ModGNN: Expert policy approximation in multi-agent systems with a modular graph neural network architecture. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE; 2021. p. 9161-7.
[110] Zhou J, Cheng J, Zhang L, Zhang W. A General Auxiliary Controller for Multi-agent Flocking. In: 2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP). IEEE; 2021. p. 789-94.
[111] Tolstaya EV, Gama F, Paulos J, Pappas G, Kumar VR, Ribeiro A. Learning Decentralized Controllers for Robot Swarms with Graph Neural Networks. ArXiv. 2019;abs/1903.10527. Available from: https://api.semanticscholar.org/CorpusID:85517736.
[112] Chen S, Sun Y, Li P, Zhou L, Lu CT. Spatial Temporal Graph Neural Networks for Decentralized Control of Robot Swarms. In: Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems. SIGSPATIAL '23. ACM; 2023. p. 1–4. Available from: http://dx.doi.org/10.1145/3589132.3625630.
[113] Subasi A. Chapter 3 - Machine learning techniques. In: Subasi A, editor. Practical Machine Learning for Data Analysis Using Python. Academic Press; 2020. p. 91-202. Available from: https://www.sciencedirect.com/science/article/pii/B9780128213797000035.
[114] Ghojogh B, Ghodsi A. Recurrent Neural Networks and Long Short-Term Memory Networks: Tutorial and

Survey; 2023. Available from: https://arxiv.org/abs/2304.11461.
[115] Song Y, Fang X, Liu B, Li C, Li Y, Yang SX. A novel foraging algorithm for swarm robotics based on virtual pheromones and neural network. Applied Soft Computing. 2020;90:106156.
[116] Tinoco CR, Oliveira G. Pherocom: decentralised and asynchronous swarm robotics coordination based on virtual pheromone and vibroacoustic communication. arXiv preprint arXiv:220213456. 2022.
[117] Le VT, Ngo TD. Virtual pheromone based network flow control for modular robotic systems. Electronics. 2020;9(3):481.
[118] Na S, Qiu Y, Turgut AE, Ulrich J, Krajník T, Yue S, et al. Bio-inspired artificial pheromone system for swarm robotics applications. Adaptive Behavior. 2021;29(4):395-415.
[119] Liu T, Sun X, Hu C, Fu Q, Yue S. A multiple pheromone communication system for swarm intelligence. IEEE Access. 2021;9:148721-37.
[120] Na S, Raoufi M, Turgut AE, Krajník T, Arvin F. Extended artificial pheromone system for swarm robotic applications. In: Artificial life conference proceedings. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ...; 2019. p. 608-15.
[121] Gosrich W, Mayya S, Li R, Paulos J, Yim M, Ribeiro A, et al.. Coverage Control in Multi-Robot Systems via Graph Neural Networks; 2021. Available from: https://arxiv.org/abs/2109.15278.
[122] Agarwal S, Ribeiro A, Kumar V. Asynchronous Perception-Action-Communication with Graph Neural Networks; 2023. Available from: https://arxiv.org/abs/2309.10164.
[123] Ghosh A, Sufian A, Sultana F, Chakrabarti A, De D. In: Fundamental Concepts of Convolutional Neural Network; 2020. p. 519-67.
[124] Muandet K, Fukumizu K, Sriperumbudur B, Schölkopf B. Kernel Mean Embedding of Distributions: A Review and Beyond. Foundations and Trends® in Machine Learning. 2017;10(1–2):1–141. Available from: http://dx.doi.org/10.1561/2200000060.
[125] Hussein A, Gaber MM, Elyan E, Jayne C. Imitation learning: A survey of learning methods. ACM Computing Surveys (CSUR). 2017;50(2):1-35.
[126] Tang C, Monteleoni C. On Lloyd's algorithm: new theoretical insights for clustering in practice. In: Artificial Intelligence and Statistics. PMLR; 2016. p. 1280-9.
[127] Faber V, Gunzburger M. Centroidal Voronoi Tessellations: Applications and Algorithms. Siam Review - SIAM REV. 1999 12;41:637-76.
[128] Agarwal S, Muthukrishnan R, Gosrich W, Kumar V, Ribeiro A. LPAC: Learnable Perception-Action-Communication Loops with Applications to Coverage Control; 2024. Available from: https://arxiv.org/abs/2401.04855.
[129] Chen S, Sun Y, Li P, Zhou L, Lu CT. Spatial Temporal Graph Neural Networks for Decentralized Control of Robot Swarms. In: Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems; 2023. p. 1-4.
[130] Chen S, Sun Y, Li P, Zhou L, Lu CT. Learning Decentralized Flocking Controllers with Spatio-Temporal Graph Neural Network. In: 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE; 2024. p. 2596-602.
[131] Liu S, Chang P, Liang W, Chakraborty N, Driggs-Campbell K. Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning. In: 2021 IEEE international conference on robotics and automation (ICRA). IEEE; 2021. p. 3517-24.
[132] Trautman P, Krause A. Unfreezing the robot: Navigation in dense, interacting crowds. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2010. p. 797-803.
[133] Long P, Liu W, Pan J. Deep-learned collision avoidance policy for distributed multiagent navigation. IEEE Robotics and Automation Letters. 2017;2(2):656-63.
[134] Golilarz NA, Gao H, Addeh A, Pirasteh S. ORCA optimization algorithm: A new meta-heuristic tool for complex optimization problems. In: 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP). IEEE; 2020. p. 198-204.
[135] Huang Z, Li R, Shin K, Driggs-Campbell K. Learning sparse interaction graphs of partially detected pedestrians for trajectory prediction. IEEE Robotics and Automation Letters. 2021;7(2):1198-205.
[136] Jang E, Gu S, Poole B. Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:161101144. 2016.
[137] Deng Z, Gao P, Jose WJ, Reardon C, Wigness M, Rogers J, et al. Coordinated multi-robot navigation with formation adaptation. In: 2025 IEEE International Conference on Robotics and Automation (ICRA). IEEE; 2025. p. 11384-91.
[138] Doya K. Reinforcement learning: Computational theory and biological mechanisms. HFSP journal. 2007;1(1):30.
[139] Mahajan S. Reinforcement learning: A review from a machine learning perspective. International Journal. 2014;4(8).
[140] Jia J, Wang W. Review of reinforcement learning research. In: 2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC). IEEE; 2020. p. 186-91.
[141] Lin B. Reinforcement learning and bandits for speech and language processing: Tutorial, review and outlook. Expert Systems with Applications. 2023:122254.
[142] Ssengonzi C, Kogeda OP, Olwal TO. A survey of deep reinforcement learning application in 5G and beyond network slicing and virtualization. Array. 2022;14:100142.
[143] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, et al. Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. PmLR; 2016. p. 1928-37.
[144] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:170706347. 2017.

[145] Schulman J, Levine S, Abbeel P, Jordan M, Moritz P. Trust region policy optimization. In: International conference on machine learning. PMLR; 2015. p. 1889-97.
[146] López-Incera A, Ried K, Müller T, Briegel HJ. Development of swarm behavior in artificial learning agents that adapt to different foraging environments. PLoS One. 2020;15(12):e0243628.
[147] Abpeikar S, Kasmarik K, Garratt M, Hunjet R, Khan MM, Qiu H. Automatic collective motion tuning using actor-critic deep reinforcement learning. Swarm and Evolutionary Computation. 2022;72:101085.
[148] Zhu W, Yu S, Chen H, Gong Z. Review of Application of Model-free Reinforcement Learning in Intelligent Decision. Highlights in Science, Engineering and Technology. 2023;56:315-23.
[149] Xie C, Patil S, Moldovan T, Levine S, Abbeel P. Model-based reinforcement learning with parametrized physical models and optimism-driven exploration. In: 2016 IEEE international conference on robotics and automation (ICRA). IEEE; 2016. p. 504-11.
[150] Kuvayev L, Sutton RS. Model-based reinforcement learning with an approximate, learned model. In: Proceedings of the ninth Yale workshop on adaptive and learning systems. Citeseer; 1996. p. 101-5.
[151] Neto G. From single-agent to multi-agent reinforcement learning: Foundational concepts and methods. Learn Theory Course 2005; 2;.
[152] Jin R, Chen Z, Lin Y, Song J, Wierman A. Approximate global convergence of independent learning in multi-agent systems. arXiv preprint arXiv:240519811. 2024.
[153] Jaquier N, et al. Transfer learning in robotics: an upcoming breakthrough. A review of promises and challenges. 2023.
[154] Yuan L, Zhang Z, Li L, Guan C, Yu Y. A survey of progress on cooperative multi-agent reinforcement learning in open environment. arXiv 2023. arXiv preprint arXiv:231201058.
[155] Liu Q, Guo J, Lin S, Ma S, Zhu J, Li Y. MASQ: Multi-Agent Reinforcement Learning for Single Quadruped Robot Locomotion. arXiv preprint arXiv:240813759. 2024.
[156] Du W, Ding S. A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications. Artificial Intelligence Review. 2021;54(5):3215-38.
[157] Zai A, Brown B. Deep reinforcement learning in action. Manning Publications; 2020.
[158] Hong D, Lee S, Cho YH, Baek D, Kim J, Chang N. Energy-efficient online path planning of multiple drones using reinforcement learning. IEEE Transactions on Vehicular Technology. 2021;70(10):9725-40.
[159] Javaid A. Understanding Dijkstra's algorithm. Available at SSRN 2340905. 2013.
[160] Wang H, Yu Y, Yuan Q. Application of Dijkstra algorithm in robot path-planning. In: 2011 second international conference on mechanic automation and control engineering. IEEE; 2011. p. 1067-9.
[161] Goh KC, Ng RB, Wong YK, Ho NJ, Chua MC. Aerial filming with synchronized drones using reinforcement learning. Multimedia Tools and Applications. 2021;80:18125-50.
[162] Venturini F, Mason F, Pase F, Chiariotti F, Testolin A, Zanella A, et al. Distributed reinforcement learning for flexible and efficient uav swarm control. IEEE Transactions on Cognitive Communications and Networking. 2021;7(3):955-69.
[163] Kozma R, Alippi C, Choe Y, Morabito FC. Artificial intelligence in the age of neural networks and brain computing. Academic Press; 2018.
[164] Ataur Khalil A, Byrne AJ, Ashiqur Rahman M, Manshaei MH. Efficient UAV Trajectory-Planning using Economic Reinforcement Learning. arXiv e-prints. 2021:arXiv-2103.
[165] Long P, Fan T, Liao X, Liu W, Zhang H, Pan J. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE; 2018. p. 6252-9.
[166] Batra S, Huang Z, Petrenko A, Kumar T, Molchanov A, Sukhatme GS. Decentralized control of quadrotor swarms with end-to-end deep reinforcement learning. In: Conference on Robot Learning. PMLR; 2022. p. 576-86.
[167] Canese L, Cardarilli GC, Di Nunzio L, Fazzolari R, Giardino D, Re M, et al. Multi-agent reinforcement learning: A review of challenges and applications. Applied Sciences. 2021;11(11):4948.
[168] Kapoor S. Multi-agent reinforcement learning: A report on challenges and approaches. arXiv preprint arXiv:180709427. 2018.
[169] Zhao X, Yang R, Zhang Y, Yan M, Yue L. Deep reinforcement learning for intelligent dual-UAV reconnaissance mission planning. Electronics. 2022;11(13):2031.
[170] Singh A, Yang L, Hartikainen K, Finn C, Levine S. End-to-end robotic reinforcement learning without reward engineering. arXiv preprint arXiv:190407854. 2019.
[171] Qi J, Zhou Q, Lei L, Zheng K. Federated reinforcement learning: Techniques, applications, and open challenges. arXiv preprint arXiv:210811887. 2021.
[172] Zhuo HH, Feng W, Lin Y, Xu Q, Yang Q. Federated deep reinforcement learning. arXiv preprint arXiv:190108277. 2019.
[173] Lee W, et al. Federated reinforcement learning-based UAV swarm system for aerial remote sensing. Wireless Communications and Mobile Computing. 2022;2022.
[174] Abpeikar S, Kasmarik K, Garratt M. Iterative transfer learning for automatic collective motion tuning on multiple robot platforms. Frontiers in Neurorobotics. 2023;17:1113991.
[175] Xu Y, Lu X, Fei Y, Huang Y. Iterative self-transfer learning: A general methodology for response time-history prediction based on small dataset. Journal of Computational Design and Engineering. 2022;9(5):2089-102.
[176] Zhong X, Guo S, Shan H, Gao L, Xue D, Zhao N. Feature-based transfer learning based on distribution similarity. IEEE Access. 2018;6:35551-7.
[177] Lowe R, Wu YI, Tamar A, Harb J, Pieter Abbeel O, Mordatch I. Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems. 2017;30.

[178] Wang Z, Guo Y, Li N, Hu S, Wang M. Autonomous collaborative combat strategy of unmanned system group in continuous dynamic environment based on PD-MADDPG. Computer Communications. 2023;200:182-204.
[179] Yin Y, Chen Z, Liu G, Yin J, Guo J. Autonomous navigation of mobile robots in unknown environments using off-policy reinforcement learning with curriculum learning. Expert Systems with Applications. 2024:123202.
[180] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. PMLR; 2018. p. 1861-70.
[181] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Advances in neural information processing systems. 2017;30.
[182] Hu K, Xu K, Xia Q, Li M, Song Z, Song L, et al. An overview: Attention mechanisms in multi-agent reinforcement learning. Neurocomputing. 2024;598:128015.
[183] Singh CD, He B, Fermüller C, Metzler C, Aloimonos Y. Minimal perception: enabling autonomy in resource-constrained robots. Frontiers in Robotics and AI. 2024;11:1431826.
[184] Abdalwhab ABM, Beltrame G, Kahou SE, St-Onge D. Attention-Based Multi-Agent RL for Multi-Machine Tending Using Mobile Robots. AI. 2025;6(10):252.
[185] Abdalwhab A, Beltrame G, Kahou SE, St-Onge D. Learning multi-agent multi-machine tending by mobile robots. arXiv preprint arXiv:240816875. 2024.
[186] Liu S, Wu Z. Efficient multi-robot exploration via multi-head attention-based cooperation strategy. arXiv preprint arXiv:191101774. 2019.
[187] Escudie E, Matignon L, Saraydaryan J. Attention graph for multi-robot social navigation with deep reinforcement learning. arXiv preprint arXiv:240117914. 2024.
[188] Jianliang A. A multimodal educational robots driven via dynamic attention. Frontiers in Neurorobotics. 2024;18:1453061.
[189] Bhuyan B, Sarma HKD, Sarma N, Kar A, Mall R. Quality of service (QoS) provisions in wireless sensor networks and related challenges. Wireless Sensor Network. 2010;2(11):861.
[190] Ming J, Xie Z, Teng H. Optimization of A comprehensive dispatching system based on ant colony algorithm and dynamic weight power dispatching strategy. Scientific Reports. 2025;15(1):39441.
[191] Liu PX, Meng M, Ye X, Gu J. An UDP-based protocol for Internet robots. In: Proceedings of the 4th World Congress on Intelligent Control and Automation (Cat. No.02EX527). vol. 1; 2002. p. 59-65 vol.1.
[192] Botta A, Rotbei S, Zinno S, Ventre G. Cyber security of robots: A comprehensive survey. Intelligent Systems with Applications. 2023;18:200237.
[193] Moroncelli A, Pacheco A, Strobel V, Lajoie PY, Dorigo M, Reina A. Byzantine fault detection in Swarm-SLAM using blockchain and geometric constraints. In: International Conference on Swarm Intelligence. Springer; 2024. p. 42-56.
[194] Yaacoub JPA, Noura HN, Salman O, Chehab A. Robotics cyber security: Vulnerabilities, attacks, countermeasures, and recommendations. International Journal of Information Security. 2022;21(1):115-58.
[195] Trujillo JC, Munguia R, Guerra E, Grau A. Visual-based SLAM configurations for cooperative multi-UAV systems with a lead agent: an observability-based approach. Sensors. 2018;18(12):4243.
[196] Trujillo JC, Munguia R, Guerra E, Grau A. Cooperative monocular-based SLAM for multi-UAV systems in GPS-denied environments. Sensors. 2018;18(5):1351.
[197] Itani M, Chen T, Yoshioka T, Gollakota S. Creating speech zones with self-distributing acoustic swarms. Nature Communications. 2023;14(1):5684.
[198] Lee S, Min BC. Distributed control of multi-robot systems in the presence of deception and denial of service attacks. arXiv preprint arXiv:210200098. 2021.
[199] Dev K, Madhwal Y, Shevelo S, Osinenko P, Yanovich Y. SwarmRaft: Leveraging Consensus for Robust Drone Swarm Coordination in GNSS-Degraded Environments. IEEE Internet of Things Journal. 2025.
[200] Chen H, Wen C, Li X. Resilient Multi-Dimensional Consensus and Distributed Optimization against Agent-Based and Denial-of-Service Attacks. arXiv preprint arXiv:251006835. 2025.
[201] Salem MA, Perez MC, Rabia AH. A TinyML Reinforcement Learning Approach for Energy-Efficient Light Control in Low-Cost Greenhouse Systems. In: 2025 Interdisciplinary Conference on Electrics and Computer (INTCEC). IEEE; 2025. p. 1-6.
[202] Alongi F, Ghielmetti N, Pau D, Terraneo F, Fornaciari W. Tiny neural networks for environmental predictions: An integrated approach with miosix. In: 2020 IEEE International Conference on Smart Computing (SMARTCOMP). IEEE; 2020. p. 350-5.
[203] Cereda E, Giusti A, Palossi D. Training on the Fly: On-device Self-supervised Learning aboard Nano-drones within 20 mW. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2024;43(11):3685-95.
[204] Amato C. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning. arXiv preprint arXiv:240903052. 2024.
[205] Attaran M, Celik BG. Digital Twin: Benefits, use cases, challenges, and opportunities. Decision Analytics Journal. 2023;6:100165.
[206] Jawhar I, Mohamed N, Wu J, Al-Jaroodi J. Networking of multi-robot systems: Architectures and requirements. Journal of Sensor and Actuator Networks. 2018;7(4):52.

I. Appendix A

Cat.	Policy name	Convergence Rate (sec)	Env Dimesion	Centralised	Distributed	Simulated	Physical	Path Planning	Task Distri	Search	Static Env	Dynamic Env	Flocking	Formation	Robots Num	Scalable
PSO variants																
	FAC-PSO [35]	11-14	2D, 3D	✓		✓		✓			✓	✓				
	HPSOGA [37]		3D	✓		✓		✓			✓			✓		
	PSO-CPTD [38]		2D	✓		✓		✓	✓		✓				6	✓
	RPSO [39]	5-10	2D, 3D	✓		✓		✓			✓	✓				✓
	IPSO [40]		2D, 3D	✓		✓		✓			✓			✓	5	✓
	MPSO [41]	20-48	2D, 3D	✓		✓	✓	✓				✓			2	✓
	HIPSO-MSOS [42]		2D, 3D	✓		✓		✓			✓					✓
	BI-PSO[43]	12	2D	✓		✓	✓	✓			✓					✓
	GA-PSO [45]		2D, 3D	✓		✓		✓	✓			✓		✓	10	✓
	VAINDIWPSO,C-VAINDIWPSO [28]		2D, 3D		✓	✓		✓			✓			✓		✓
	SDPSO [46]		3D	✓		✓		✓			✓					
GWO variants	PSO-GA-DWPA[67]		2D	✓		✓			✓		✓				100	✓
	MPPWPA [69]		2D	✓		✓			✓		✓				100	✓
	IGWO [70]		2D	✓		✓		✓			✓					✓
	[71]		2D		✓	✓		✓	✓	✓	✓				4	✓
	GWO–WOA[75]		3D	✓		✓	✓				✓				2	
	IA, GWO, VNS[78]		2D	✓		✓				✓						
ACO variants	V-ACO [52]		3D	✓		✓		✓			✓			✓		
	MC-ACO [53]		3D		✓	✓		✓			✓				3	✓
	Improved ACO [54]					✓				✓		✓				
	CACOC [55]		2D, 3D		✓	✓		✓		✓	✓				10	✓
	CACOC[57]		2D, 3D		✓	✓		✓		✓		✓			10	✓
	CACOC+CA [59]		2D		✓	✓				✓	✓				3	✓
	MPC-CA [60]		2D, 3D	✓								✓				
	CACOC, ABISS [61]		2D		✓	✓				✓	✓				6	✓
	CACOC [62]		2D		✓	✓				✓	✓				10	

continued on next page

Table 3 – *continued from previous page*

Cat.	Policy name	Convergence Rate (sec)	Env Dimesion	Centralised	Distributed	Simulated	Physical	Path Planning	Task Distri	Search	Static Env	Dynamic Env	Flocking	Formation	Robots Num	Scalable
	ACS-VND, MMAS-VND [63]		2D	✓		✓		✓	✓		✓				38	✓
	IHMACO [66]		2D	✓		✓		✓			✓				80	✓
SPP Algorithms	DFM [83]	3.2	2D		✓	✓						✓	✓		7	
	SIMFC [84]		2D, 3D		✓	✓	✓					✓	✓		4	✓
	[85]		2D		✓	✓	✓				✓		✓		10	✓
	DFFFC[87]	1.5	2D		✓	✓	✓	✓				✓	✓	✓	11	✓
	SPFA[7]	1	3D		✓	✓	✓				✓	✓	✓		9	✓
Learning Based	ModGNN [109]		3D		✓	✓						✓	✓		64	✓
	Aux Controller[110]		3D		✓	✓					✓		✓		100	✓
	ST-GNN [112]		2D		✓	✓						✓	✓		100	✓
	DWENN [115]		2D		✓	✓			✓	✓	✓	✓			20	✓
	APAC [122]		2D		✓	✓		✓			✓				32	✓
	LPAC [128]		2D		✓	✓					✓				64	✓
	ST-GNN [129, 130]		3D		✓	✓	✓				✓		✓		6	✓
	TD3 [158]		3D		✓	✓		✓				✓				✓
	DRQN [161]		3D	✓	✓	✓						✓		✓	4	
	DQN [162]		2D		✓	✓				✓	✓				16	✓
	REPlanner [164]		2D		✓	✓		✓	✓		✓				3	✓
	PPO end-to-end RL policy[166]		3D		✓	✓	✓					✓	✓	✓	8	✓
	PPO-based end-to-end planning [169]		2D	✓		✓		✓			✓				2	✓
	FRL based UAV Swarm[173]		2D		✓	✓				✓		✓			4	✓
	ILCoMoT[174]	10	2D	✓		✓	✓				✓		✓		3	
	PD-MADDPG [178]		2D		✓	✓						✓			5	
	SCF [179]		2D	✓		✓	✓	✓			✓	✓				
	AB-MAPPO [184, 185]		2D		✓	✓		✓	✓			✓			3	

continued on next page

Table 3 – *continued from previous page*

Cat.	Policy name	Convergence Rate (sec)	Env Dimesion	Centralised	Distributed	Simulated	Physical	Path Planning	Task Distri	Search	Static Env	Dynamic Env	Flocking	Formation	Robots Num	Scalable
	MAMECS [186]		2D		✓	✓				✓		✓			3	
	MultiSoc[187]		2D		✓	✓		✓				✓			20	✓
	DRLMACA[165]	43,200	2D		✓	✓		✓				✓			100	✓
	Res-ALBEF [188]		2D,3D			✓		✓								✓

Table 3. Comparison of swarm intelligence approaches in the literature.

Key: ✓ = present; blank = not reported.

II. Appendix B

Table 4. Wireless network protocols for MRS systems [206].

Protocol	**Data Rate**	**Transmission Range**	**MRS System Communication and Link Types**
IEEE 802.15.1 (Bluetooth)	1 to 24 Mbps	10 m	Short-range Robot to Robot and Robot to Infrustructure communication.
IEEE 802.15.3	11 to 55 Mbps	10 m	High-rate short-range Robot to Robot and Robot to Infrustructure communication.
IEEE 802.11a/b/g/n/ac	15, 30, 45, 60, 90, 120, 135, 150, 346, 800, 3466 Mbps	Up to 250 m	Medium range Robot to Robot and Robot to Infrustructure communication.
IEEE 802.16/rel 1/rel 1.5/rel 2(m) (WiMAX)	2 to 75 Mbps	Up to 35 56 km	Long range Robot to Robot and Robot to Infrustructure communication
Cellular 3G	144 Kbps (mobile users) to 42 Mbps (for stationary users)	Cell radius dependent, up to several km's	Mostly Robot to Infrustructure communication.
Cellular 4G/LTE	300 Mbps to 1 Gbps	up to several km's	Mostly Robot to Infrustructure communication.
Satellite (LEO/MEO/GEO)	10 Mbps (upload) and 1 Gbps (download)	Covers hundreds of km's to entire earth	Mostly Robot to Infrustructure communication.