UDC 004.8:681.5:629.735
DOI: 10.20535/SRIT.2308-8893.2025.3.11

METHODS OF SWARM ARTIFICIAL INTELLIGENCE IN AUTONOMOUS NAVIGATION TASKS OF UAVS

M.Z. ZGUROVSKY, Yu.P. ZAYCHENKO, A.M. TYTARENKO, O.V. KUZMENKO

Abstract. This paper presents a comparative analysis of nine swarm intelligence (SI) methods in terms of their suitability for onboard AI platforms in autonomous unmanned aerial vehicle (UAV) swarms. A set of key criteria is defined, including computational complexity, scalability, latency, robustness to agent loss, and adaptability. Decentralized Behavior Trees (BTs) are identified as the most balanced approach for the reactive behavior layer, while the global swarm optimization method GBestPSO proves effective for high-level planning. A hybrid two-layer cognitive architecture is proposed that integrates BTs and GBestPSO, with functional separation between layers and communication based on DDS/RTPS protocols. The architecture exhibits high autonomy, fault tolerance, modularity, and suitability for real-time embedded systems operating in dynamic or adversarial environments. The results were partially supported by the National Research Foundation of Ukraine, grant No. 2025.06/0022 "AI platform with cognitive services for coordinated autonomous navigation of distributed systems consisting of a large number of objects".

Keywords: swarm intelligence, UAV, autonomous navigation, behavior trees, GBestPSO, ROS 2, DDS, cognitive architecture.

INTRODUCTION

In the current environment of increasing complexity and dynamism in both military and civilian settings, autonomous swarms of unmanned aerial vehicles (UAVs) are increasingly being seen as an effective tool for reconnaissance, patrol, escort, and rapid response missions [1; 2]. Their advantages include high mobility, the ability to cover large areas, and the ability to operate in a decentralized mode [3]. However, to achieve true autonomy, each agent in the swarm must have the cognitive ability to perceive the environment, assess the situation, predict the consequences of its actions, and interact with other agents without centralized control [4].

Building such a system requires the deployment of an onboard AI platform with cognitive services capable of functioning in conditions of partial or complete loss of communication, in an informationally complex environment, or under the influence of electronic warfare [5]. Architecturally, this platform includes several functional components: a cognitive core, a sensory-analytical layer, a navigation controller, a communication module, a security module, a digital twin, and recon-

figuration services [6]. The cognitive core plays a decisive role in this architecture, implementing the agent's intellectual subjectivity. It is based on the swarm artificial intelligence (AI) method, which forms mechanisms of perception, planning, coordination, and adaptation [7].

In this context, a key scientific and practical task arises – the choice of a swarm AI method suitable for implementation on board a drone. Such a method must meet strict limitations on computing resources (limited processor performance, small memory capacity), ensure real-time operation (low decision-making latency), support scaling to tens or hundreds of agents, be resistant to swarm element losses, and be adaptive to dynamic environmental changes [8].

In addition, the method must be integrated into the open-source modular platform Robot Operating System (ROS) using the Data Distribution Service (DDS) [9; 10], and must support working with simulation digital twins, which are critical for validating and training behavioural models in a safe virtual environment [11].

This challenge has no trivial solution. Traditional approaches — such as centralized planning, ant algorithms, and reinforcement learning methods — although highly effective in certain aspects, are usually overly resource-intensive, dependent on stable connectivity, or too inert to adapt to environmental changes [12; 13].

Therefore, this research focuses on finding, comparing, and justifying the most acceptable swarm AI method for embedding into an onboard AI platform with cognitive services, capable of not only ensuring the autonomous behaviour of an individual drone, but also implementing a holistic, adaptive, self-reconfiguring swarm system of a new generation.

CRITERIA FOR THE CHOICE OF SWARM METHODS OF AN ON-BOARD AI PLATFORM WITH COGNITIVE SERVICES

The successful implementation of autonomous swarm navigation for drones requires careful selection of a swarm artificial intelligence method that not only provides the necessary behavioural complexity but also meets the limitations of the onboard AI platform. The basis for this selection is a set of criteria that consider both the technical and functional requirements for an autonomous multin-drone system. Below is a list of key criteria and their rationale:

1. **Computational complexity.** This criterion assesses how much the swarm AI method loads the processor when the number of agents increases. Quadratic or cubic complexity (e.g., $O(n^2)$, $O(n^3)$) significantly limits scalability and performance when running on embedded processors, especially in environments with limited computing resources (e.g., STM32, Raspberry Pi CM4, Jetson Nano). Methods with linear or logarithmic complexity ($O(n)$, $O(n \log n)$) scale better and are more suitable for decentralized autonomy [8; 18].

2. **Memory Requirements.** Methods that require large buffers, tables, and historical data are not suitable for implementation on compact boards without additional GPU or expanded memory. The total amount of memory required to store internal variables, sensor maps, behavioural patterns, etc. is considered [6].

3. **Latency.** Latency is critical for real-time applications: the system must respond to changes in the environment or commands with a delay of no more than

50–70 ms. Methods with high latency do not allow the drone to manoeuvre safely or make timely decisions in combat or rescue operations [13].

4. **Scalability.** This indicator determines how effectively the method maintains performance as the number of agents increases. For example, the Boids method scales well to 100+ agents, while ACO scales to no more than 30. High scalability is a prerequisite for complex scenarios with dozens or hundreds of drones [3].

5. **Robustness to Agent Loss.** In military and unstable environments, drones may be lost. The swarm AI method must automatically adapt to a decrease in the number of swarm members. Centralized approaches or Leader-Follower models are vulnerable to such losses, while decentralized behaviour trees (BTs) remain functional even when individual nodes are lost [17].

6. **Environmental Adaptivity.** True autonomy requires the ability to respond to changes in the environment: obstacles, changes in terrain, dynamic threats. Methods with low levels of sensory integration (e.g., classical PSO) have limited adaptability. In contrast, behavioural trees or DRL with sensory context integration are capable of dynamically reconfiguring behaviour [14].

7. **Onboard Real-Time Suitability.** This criterion refers to the ability of the algorithm to operate without the need for an external computing centre or cloud services. The method must function autonomously on the drone's hardware platform, considering real time, energy efficiency, and resource limitations [9]. Algorithms that already have known libraries under ROS 2, RTOS, or support DDS protocols are particularly valued.

These seven criteria form the basis of a multi-criteria model for evaluating and selecting a swarm AI method. They not only formalize the decision-making process, but also directly influence the architecture of the AI platform, its stability, adaptability, and viability in a real-world environment.

ANALYSIS AND SELECTION OF SWARM METHODS FOR USE IN AN AI PLATFORM WITH COGNITIVE SERVICES

Developing an effective AI platform for an autonomous swarm of unmanned aerial vehicles involves choosing a swarm method that not only meets functional requirements (adaptation, coordination, decentralization) but also complies with the strict limitations of the real environment: limited computing resources, the need for real-time decision-making, loss of communication, or individual agents. That is why it is crucial to compare existing methods in terms of computational complexity, memory requirements, latency, scalability, resilience to agent loss, adaptability, and suitability for onboard implementation (Table 1).

Classic swarm optimization methods, particularly Method 1 and its global modification (Method 9), demonstrate moderate complexity ($O(n^2)$) and scale well to 50 agents. They are suitable for search and global positioning tasks, but depend on one or more leader agents, which is a vulnerability in real-world environments.

Method 2, on the other hand, shows better adaptation to complex environments through a pheromone amplification mechanism, but has significantly higher complexity and latency, which makes it unsuitable for real-time tasks on typical onboard processors.

Both methods (1 and 2) are suitable for centralized or periodic optimization tasks but are limited in scenarios where continuous adaptation to the external environment is required.

Methods 3 and 5 are characterized by the lowest computational complexity ($O(n)$), high scalability (100+ agents), and low latency (< 30 ms), which makes them technically attractive. However, they exhibit limited cognitive ability: Boids does not support goal-oriented planning, and stigmergy requires precise signal tuning and has limited adaptation to unpredictable scenarios.

Table 1. Comparative table of swarm AI methods

N	Swarm AI method	Computational complexity	Memory Requirements	Latency	Scalability	Robustness to Agent Loss	Environmental Adaptivity	Onboard Real-Time Suitability
1	Particle Swarm Optimization (PSO) [16]	Medium ($O(n^2)$)	Medium	< 50 ms	Good (up to ~50 agents)	Medium	Limited	Yes
2	Ant Colony Optimization (ACO) [3]	High ($O(n^2)$ log n)	High	100–200 ms	Limited (up to 30 agents)	High	Good	Limited
3	Boids (Reynolds' Rules) [21]	Low ($O(n)$)	Low	< 20 ms	High (100 + agents)	Low	Low	Yes
4	Consensus-based Algorithms [19]	Medium ($O(n \log n)$)	Medium	50–100 ms	Medium (up to 50 agents)	Medium	Medium	Yes
5	Stigmergy-based Models [20]	Low ($O(n)$)	Low	< 30 ms	High (100+ agents)	High	Good	Yes
6	Deep Reinforcement Learning (DRL) [22]	High ($O(n^3)$)	High	> 200 ms	Limited (20–30 agents)	Medium	High	No
7	Decentralized Behaviour Trees (BTs) [18]	Medium ($O(n \log n)$)	Medium	30–70 ms	High (50–100 agents)	High	High	No
8	Leader-Follower [17]	Low ($O(n)$)	Low	< 20 ms	Limited (~10–20 drones in classic implementation)	Low–medium (loss of leader → critical risk)	Limited (mainly in modified versions)	Yes, especially with low data intensity
9	Global swarm optimization method (GBestPSO) [16,23]	Medium ($O(n^2)$)	Medium	< 50 ms	Good (up to 50 agents)	Medium (dependence on leader)	Limited	Limited

Consensus-based algorithms (Method 5) ensure consistency within the swarm with moderate complexity and delays. However, they also do not allow modeling complex agent behavior or changing the mission structure during its execution. In turn, the Leader-Follower method, despite its simple implementation and low latency, has a fatal flaw – critical dependence on the leader. If the leader is lost, coordinated behavior is destroyed, which is unacceptable for combat or emergency scenarios.

The deep reinforcement learning method (Method 6), on the contrary, demonstrates the highest level of adaptability: models can self-learn and generalize knowledge about new environments. However, the inference delay exceeds 200

ms even on easy tasks, and the hardware requirements include a GPU (Jetson Xavier, Orin, or Nvidia RTX), which is beyond the capabilities of typical onboard platforms.

The most balanced method in terms of all criteria is the decentralized behavior tree method (Method 7). It has moderate computational complexity ($O(n \log n)$), operates with an average latency of 30–70 ms even on weak processors, scales up to 100 agents, demonstrates high adaptability, resilience to losses, and is suitable for implementation in ROS 2 with DDS protocols. BTs provides autonomous decision-making logic for each agent based on local context, sensor information, and a predefined behavior structure. In addition, it supports tree reconfiguration during a mission, which is especially important for tasks with variable goal or role structures in a swarm system.

Therefore, although Method 7 is the best choice for building the basic cognitive architecture of the agent, it is advisable to use a hybrid approach in which the global swarm optimization method (Method 9) plays a supporting role in global or semi-global optimization tasks. This division of functions allows BTs to be responsible for reactive behavior, real-time decision-making, and fault tolerance, while GBestPSO is responsible for strategic tasks, such as finding optimal agent locations, distributing subtasks, or optimizing configuration at the beginning of a mission.

This combination allows for the implementation of a two-level cognitive architecture: behavioral level – BTs, planning level – GBestPSO. Interaction between these levels can be carried out via DDS messages, which will ensure deterministic exchange between autonomous agents without loss of synchronization.

Figure 1 illustrates the two-level cognitive architecture of an autonomous swarm system that combines two complementary approaches: behavior trees (BTs) at the behavioral level and global swarm optimization GBestPSO at the planning level. The architecture is designed to ensure autonomous, adaptive, and fault-tolerant behavior of the UAV swarm in dynamic environments and in the event of communication loss.

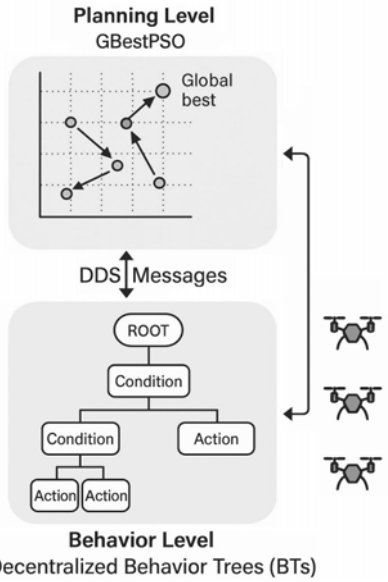

Fig. 1. Two-level cognitive architecture: planning level — GBestPSO, behavioral level — BTs

The lower block of the figure reflects the logic of the behavioral level, constructed in the form of a decision tree. The ROOT node initiates the execution of the tree, CONDITION defines the situational conditions (for example, the presence of an obstacle or loss of communication), and ACTION represents the drone's reactive actions in response to these conditions. This structure allows each agent to make decisions autonomously based on the local context. This level is implemented locally on board the drone and provides the ability to react immediately, dynamically reconfigure the tree, and operate in real time.

The upper block represents the planning level — global swarm optimization using the GBestPSO method, in which agents

collectively search for optimal locations or configurations. Here, GLOBAL BEST means the best position found by all agents, which is transmitted to others as a strategic reference point. This benchmark is transmitted to the behavioral level in the form of mission conditions via DDS/RTPS protocols. In case of communication loss, the planning level can be disabled, and the behavioral level will continue to operate using the last received gBest.

The communication between levels is provided through DDS messages, which guarantees reliable, deterministic communication without centralized control. This integration allows for a flexible balance between the local autonomy of each drone and the global coordination of the entire swarm, ensuring high system efficiency in complex scenarios, including military or rescue operations.

Therefore, the proposed hybrid approach combines the modularity, adaptability, and speed of BTs with the global optimization capabilities of GBestPSO, creating a flexible, scalable, and technically feasible next-generation AI platform for UAV swarms.

THE BASIC ARCHITECTURE OF THE HYBRID APPLICATION OF METHODS BTS + GBESTPSO

To respond to the challenges of autonomous swarm navigation in conditions of limited resources and a highly dynamic environment, a hybrid two-level architecture has been proposed that combines the declarative-reactive approach of behavioral trees (BTs) with the evolutionary-optimization strategy of the global particle swarm (GBestPSO). Such integration allows for the simultaneous achievement of real-time performance, adaptability, scalability, and initial strategic coordination of swarm behavior.

The structure of hybrid architecture is divided into two functional levels:

- **Planning level (GBestPSO-Level):** implements global or semi-global optimization at the beginning of the mission or periodically in task replanning mode. It determines the selection of scenarios for current tasks, the strategic positions of agents, task clustering, and the distribution of roles or target points.
- **Behavioral level (BT-Level):** provides reactive decision-making by each agent in real time [18], including analysis of the local context, avoidance of obstacles, adaptation to role changes, loss of communication, or loss of an agent.

Communication between levels is carried out via DDS/RTPS protocols [9] in the form of semantically described messages (Table 2).

Data exchange between subsystems of two levels is organized via DDS/RTPS protocols [9], which allow semantically structured messages to be transmitted between agents. For example, the Swarm Optimizer module, which operates at the GBestPSO level, periodically calculates the globally best swarm location strategy (gBest) and transmits it via DDS as an optimized configuration or task. At the behavioral level, such information is interpreted as external mission conditions, which are automatically reflected in the updated behavioral logic through the Condition → Action tree branches. Importantly, even if communication with the planning level is lost, the behavioral level continues to autonomously execute the mission based on the last received strategic guideline. Mutual adaptation is ensured by QoS DDS channels, which provide priority delivery of important messages, such as changes in behavior tree statuses, sensor events, or strategic instructions. The above-mentioned relationships between modules can be represented by the following list:

Table 2. Main modules and functions of the basic hybrid architecture 7. BTs + 9. GBestPSO

Level	Module	Functions
BT level	Behavior Manager	Interprets behavior of trees in real time, activates local actions, reconfigures the tree in response to external changes
	Perception Module	Processing sensor data, recognizing situations, building a local map
	Communication Module	DDS/RTPS exchange of states, positions, parts of the BT tree
	Local Planner	Low-level planning (SLAM, obstacle avoidance, actions)
	Adaptation Module	Dynamic tree reconfiguration in case of environmental changes or losses
GBestPSO level	Swarm Optimizer	Implements a global PSO algorithm; forms a gBest position that sets a strategic landmark
	Global Task Allocator	Distribution of subtasks, goals, or roles among agents based on the gBest position
	Mission Planner (optional)	Scenario planning for complex missions or dynamic restarts
Both levels	Telemetry and Diagnostics	Status exchange, logging, communication with ground station

- The Swarm Optimizer module (at the GBestPSO level) periodically calculates the global best strategy (gBest) and transmits it via DDS in the form of tasks or configurations.
- The behavioral level (BT-Level) perceives these tasks as "external mission conditions" that are reflected in the corresponding branches of the behavior tree.
- Change in **gBest** → change in agent behavior through the reaction of **Condition** → **Action** nodes.
- Reaction to loss of communication: BT-Level works autonomously, with the last accepted gBest, allowing the agent to continue the mission without external optimization.
- Mutual adaptation is provided through QoS DDS channels, which prioritize behavioral commands, sensor events, tree statuses, and strategic instructions.

The software and hardware implementation of this architecture is based on ROS 2 (Foxy or Humble versions) and RTOS for real-time systems such as STM32H7. The communication layer is implemented using Fast DDS [9] or Cyclone DDS with QoS support. At the library level, BehaviorTree.CPP is used for BT modules and our own implementation of the GBestPSO algorithm based on the classical approach [14]. Jetson Orin Nano, Xavier NX, Raspberry Pi CM4 with Coral TPU, as well as lightweight STM32H7 controllers are used as hardware platforms. The main components of the software and hardware architecture are:

- Operating system: ROS 2 (Foxy/Humble), RTOS (on STM32H7).
- Communication: Fast DDS / Cyclone DDS (with QoS support).
- Libraries: BehaviorTree.CPP (BT-level), own implementation of GBestPSO (based on [14]).
- Platforms: Jetson Orin Nano / Xavier NX, Raspberry Pi CM4 with Coral TPU, STM32H7.

Among the advantages of the proposed architecture, its autonomy, flexibility, and fault tolerance should be noted. The behavioral level ensures local adaptability and the ability to respond to changes in the environment without the need for global control. The planning level, in turn, ensures strategic coordination be-

tween agents, forming a single target configuration of the system. The use of DDS protocols eliminates dependence on a central node, which significantly increases fault tolerance. In the event of a loss of the global reference point (gBest), the system automatically switches to fully local mode without losing functionality. The modular structure of this architecture ensures its scalability and adaptability to different mission classes: from reconnaissance and escort to combat or search and rescue operations. Thus, the main advantages of hybrid architecture are:

- BT-Level provides local adaptability and autonomy.
- GBestPSO-Level adds strategic coordination and global coordination.
- Interconnection via DDS eliminates dependence on a central node.
- High fault tolerance: if gBest is lost, the architecture switches to fully local mode.
- Modularity support allows architecture to be scaled for different mission classes (reconnaissance, escort, attack).

Thus, the proposed hybrid architecture, combining behavioral trees (BTs) and global particle swarm optimization (GBestPSO), demonstrates an effective combination of local autonomy and strategic coordination in swarm navigation tasks. Its two-level structure allows it to simultaneously achieve adaptability, responsiveness, and energy efficiency while maintaining a high degree of resilience to failures and communication losses. The behavioral level provides an immediate response to environmental dynamics, while the planning level defines global landmarks and roles, which increases the integrity of swarm behavior in complex conditions. The use of DDS/RTPS protocols allows for reliable, distributed communication between agents without centralized dependency. Such architecture not only has high application potential in the military and civilian spheres but also provides a scalable foundation for the further evolution of swarm intelligence systems towards the self-learning and self-coordinated platforms of the future.

ALGORITHM

The movement of each drone is described by two main phases:

1. **Phase 1: Approach to the Target.** Drones move toward target Y^* according to the velocity equation with the local best position. The influence of cognitive (C_1) and social (C_2) components depends on the distance to the leader (Y_L) and target, respectively:

$$v_{ij}(t+1) = v_{ij}(t) + C_1(d(Y_L(t), X_i(t)))r_1(t)[Y_{L_j}(t) - X_{ij}(t)] + \\ + C_2(d(Y^*(t)X_i(t)))r_2(t)[Y_j^*(t) - X_{ij}(t)], \quad (1)$$

where coefficients C_1 and C_2 are linearly dependent on distance, which allows adjusting the influence of the leader and the target on the behavior of the swarm:

$$C_1(d(Y_L(t), X_i(t))) = \frac{C_{1\max} - C_{1\min}}{Y_L(0) - X_i(0)}(Y_{Lj}(t) - X_{ij}(t)) + C_{1\min}; \quad (2)$$

$$C_2(d(Y^*(t), X_i(t))) = \frac{C_{2\max} - C_{2\min}}{Y^*(0) - X_i(0)}(Y_j^*(t) - X_{ij}(t)) + C_{2\min}. \quad (3)$$

2. **Phase 2: Firing Ring for Killing.** After reaching the target's circumference with a radius of r_0, the drones switch to polar coordinates and begin to rotate

around the target. The coordinates of the leader Y_L are described using the radius r_0 and angle $\varphi_L(t)$:

$$\begin{cases} y_1^L(t) = r_0 \cos(\varphi_L(t)), \\ y_2^L(t) = r_0 \sin(\varphi_L(t)). \end{cases} \quad (4)$$

The angular position is updated using the following formula:

$$\varphi_L(t+1) = \varphi_L(t) + \omega_L(t)t, \quad (5)$$

where $\omega_L(t)$ — angular velocity of rotation.

Each drone is controlled by its own BT, which cyclically processes the tick. The tree consists of control nodes (Selector, Sequence) and execution nodes (Condition, Action). Behavior Tree diagram:

- **Root:** initiates tree execution.
- **Sequence:** main sequence of actions for each drone.
- **Parallel:** used for simultaneous execution of tasks independent of the main movement cycle, such as modeling the impact of electronic warfare and drone loss.
  - **Action: Model_Jamming()** — simulates the probability of communication loss (P_{10}).
- **Action: Model_Drone_Loss()** — simulates the probability of drone loss (P_2).
- **Selector (Main mission logic):** Checks the mission status and switches between phases.
  - **Branch 1: Approach to the Target**.
    - **Condition: Is_Far_From_Target(r_0)** – checks whether the distance to the target is $> r_0$.
    - **Action: Run_LBestCombinedPSO_Approach_Phase()** – calculates the new speed and position.
    - **Action: Elect_New_Leader()** – this node is activated in case of Failure of the **Is_Leader_Connected()** node, which implements the self-organization mechanism.
  - Branch 2: **Firing Ring for Killing.**
    - **Condition: Is_Close_To_Target(r_0)** – checks whether the distance to the target is $\le r_0$.
    - **Sequence: Firing_Ring_Sequence.**
      - **Action: Convert_To_Polar()** – transition to polar coordinates.
      - **Decorator: Sync_Attack()** – uses the Decorator
      - **Decorator: Sync_Attack()** – uses the Decorator mechanism for synchronization. The node allows the execution of the child node only after $|ready\_agents| \ge M_{Y^*}$.
      - **Action: Attack_Target()** – attack the target.

A key element of the system is its ability to be adaptive:

- **Loss of leader.** If the leader drone loses connection with other agents (simulated by the **Model_Jamming** node), this initiates **Failure** in the leader verification conditions. The **Selector** in BT automatically switches to the branch that triggers self-organization. A new leader is selected based on the criterion $Y_{Lnew}(t) = X_i(t)$, where $d(X_i(t), Y^*) = \min_i d(X_i(t), Y^*)$.

Synchronization. The **Decorator** node ensures that the ring firing phase does ot start until enough drones (M_Y) reach the target area, which is critical for coordinating the attack.

Inertial component. In the equation, the velocity $v_{ij}(t)$ acts as an inertial component, preventing sudden changes in trajectory and ensuring smooth movement.

EXAMPLE

Let us consider a practical simulation experiment using a two-level cognitive architecture consisting of a group of drones and integrating behavioral trees (BTs) and global optimization using the GBestPSO method. According to this experiment:

- **The behavioral level (BT-level)** functioned locally on board each drone and provided reactive adaptation to sensory events such as obstacle detection, role changes, or loss of communication. This level was implemented using the BehaviorTree.CPP library with a built-in dynamic tree reconfiguration mechanism. Condition → Action nodes allowed for the rapid transformation of behavioral logic in response to changes in external mission conditions received from the planning level.
- **The planning level based on GBestPSO** was responsible for forming the strategic configuration of the swarm through global or local optimization. It periodically calculated the globally best position or scenario (gBest), which was broadcast as strategic guidelines to each agent. Data transfer between levels was carried out using DDS/RTPS protocols, which guaranteed quality of service (QoS), buffering of critical messages, and resistance to temporary communication losses.
- **The simulation experiment** was conducted in a configuration of 5–10 drones equipped with Jetson Orin Nano or Raspberry Pi CM4 paired with STM32H7 (Table 3).

Table 3. Quantitative characteristics of the simulation experiment

No	Parameter	Value
1	Number of drones in the swarm group	5–10 (Jetson Orin Nano or Raspberry Pi CM4 + STM32H7)
2	Coverage area	100 × 100 m
3	Number of control points	6
4	Mission type	Search and patrol with reconfiguration elements
5	BTs algorithm	6–10 conditional nodes, 3–4 adaptive branches
6	GBestPSO algorithm	Classic method [23]
7	Number of PSO iterations before start	30
8	BT module response delay	~45 ms (on Jetson Nano)
9	gBest transmission delay via DDS	~10–15 ms
10	Tree adaptation time when changing gBest	< 150 ms
11	Simulated communication losses	up to 30% of DDS packets
12	Mission success rate	> 95% (all drones completed the route or selected fallback actions)

The mission covered an area of 100 × 100 m, contained six control points, and was of a search-and-patrol nature with elements of reconfiguration. The BTs algorithm included 6 to 10 conditional nodes and 3–4 adaptive branches responsible for changing behavior in conditions of uncertainty. The GBestPSO algorithm was implemented in its classical form [23] with typical parameters: space dimen-

sion $d = 2$, inertial weight $\omega = 0.8$, attraction coefficients $C_1 = C_2 = 1.497$. Before the start of the mission, 30 PSO iterations were performed to find the initial strategic configuration. The BT module's response delay to an event was about 45 ms, gBest transmission via DDS was 10–15 ms, and behavior tree adaptation when gBest changed took up to 150 ms. Even with a simulated loss of up to 30% of DDS messages, the mission success rate exceeded95%.

In critical situations, the system demonstrated high adaptability. In the event of a single drone failure, other agents automatically restructured the behavior tree to compensate for the loss. If the planning level lost communication, the BT modules continued to operate autonomously based on the last strategic reference point. When obstacles were detected, the SLAM module formed a new route, and the corresponding branch of the behavior tree activated avoidance actions. The total mission time exceeded the baseline scenario by no more than 12%. The average delay between events and response was about 90 ms. The positioning error in areas without GNSS remained within 2.8 m, and swarm synchronization was maintained even with the loss of up to 40% of DDS messages.

The diagram (Fig. 2) shows a typical scenario for the implementation of a UAV swarm mission for the proposed architecture with hybrid integration of BTs and GBestPSO. This implementation is viable and technically effective for complex search and patrol missions in conditions of limited communication and a dynamic environment.

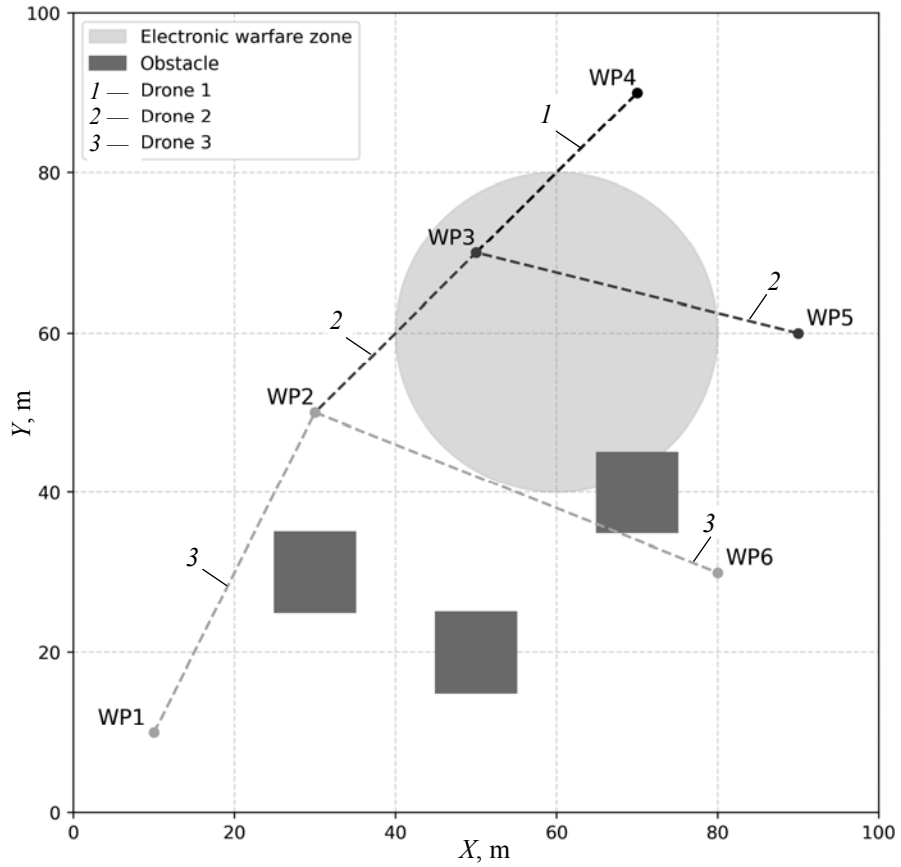

Fig. 2. Scenario for implementing the mission of a swarm of UAVs for the hybrid architecture of BTs and GBestPSO

The following images are used in Fig. 2: blue dots (WP1–WP6) – mission route control points; red circle – electronic warfare (EW) zone; gray squares — obstacles on the route (buildings, infrastructure objects); lines with markers — trajectories of three drones moving autonomously using behavioral trees (BTs), responding to obstacles and conditions broadcast from the global level of GBestPSO.

This example demonstrates the realistic implementation of a two-level cognitive architecture for swarm drones in a simulation environment with digital twin elements under threatening conditions and limited connectivity. It confirms the feasibility of the BTs + GBestPSO hybrid approach in complex, dynamic, or hostile conditions.

CONCLUSIONS

1. The article analyzes modern swarm artificial intelligence methods from the perspective of their suitability for implementation as part of an onboard AI platform for autonomous unmanned aerial vehicles. Considering the requirements for limited computing resources, real-time operation, resistance to agent loss, and adaptability, a list of key criteria for selecting a swarm AI method was formulated.

2. A comparative analysis of nine leading methods allowed us to identify decentralized behavioral trees (BTs) as the most balanced approach for the basic cognitive architecture of an agent. BTs combine low latency, resilience to losses, adaptability to the environment, and suitability for onboard implementation in ROS 2. At the same time, the GBestPSO method, as a classic global swarm optimization tool, proved to be useful for performing strategic tasks at the higher level of the system, for the initial configuration of the swarm, task distribution, and search for optimal agent locations.

3. The hybrid architecture proposed in the article, which combines BTs and GBestPSO into a two-level structure, allows for a compromise between local responsiveness and global coordination. This architecture is characterized by high autonomy, fault tolerance, scalability, and real-time adaptability. It provides a flexible division of responsibilities: BTs for immediate behavioral response, GBestPSO for planning optimization. The communication foundation, implemented through DDS/RTPS, eliminates dependence on central control and guarantees deterministic data exchange between agents.

4. A practical simulation experiment using digital twins and the BTs + GBestPSO hybrid architecture demonstrated its effectiveness in complex dynamic conditions. Even with the loss of up to 30% of DDS network messages, the system maintained coordinated swarm behavior, and the response time between a sensor event and tree restructuring averaged ~90 ms. Adaptation to new obstacles, role changes, resistance to agent failures, and the ability to operate without a global reference point demonstrated the high viability of the architecture. Mission success exceeded 95%, confirming the practical feasibility of the proposed approach.

5. Thus, the results of the study not only confirm the theoretical validity of hybrid architecture but also demonstrate its real implementation in conditions close to combat or emergency rescue situations. The proposed system can serve as a basis for building a new generation of swarm AI platforms with the potential for further evolution towards self-learning, self-coordinated, and safe-to-use autonomous systems in real environments.

REFERENCES

1. D. Floreano, R.J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, 521(7553), pp. 460–466, 2015. doi: https://doi.org/10.1038/nature14542
2. M.J. Krieger, J.-B. Billeter, "The call of duty: Self-organised task allocation in a population of up to twelve mobile robots," *Robotics and Autonomous Systems*, 30(1–2), pp. 65–84, 2000. doi: 10.1016/S0921-8890(99)00065-2
3. M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, 7, pp. 1–41, 2013. doi: 10.1007/s11721-012-0075-2
4. G.-Z. Yang et al., "The grand challenges of Science Robotics," *Science Robotics*, 3(14), eaar7650, 2018. doi: 10.1126/scirobotics.aar7650
5. Y. Mekdad, A. Arış, L. Babun, A. El Fergougui, M. Conti, R. Lazzeretti, S. Uluagac, "A survey on security and privacy issues of UAVs," *Computer Networks*, vol. 224, 109626, 2023. doi: 10.1016/j.comnet.2023.109626
6. B. Barricelli, E. Casiraghi, D. Fogli, "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications," *IEEE Access*, pp. 167653–167671, 2019. doi: 10.1109/ACCESS.2019.2953499
7. V. Trianni, A. Campo, *Fundamental Collective Behaviors in Swarm Robotics*, pp. 1377–1394, 2015. doi: 10.1007/978-3-662-43505-2_71
8. H. Hamann, *Swarm Robotics: A Formal Approach.* Springer, Cham, 2018. doi: https://doi.org/10.1007/978-3-319-74528-2
9. Y. Maruyama, S. Kato, T. Azumi, *Exploring the performance of ROS2*, pp. 1–10, 2016. doi: 10.1145/2968478.2968502
10. "Data Distribution Service (DDS) Specification. Version 1.4.", *Object Management Group (OMG)*. 2015. Available: https://www.omg.org/spec/DDS
11. S. Boschert, R. Rosen, "Digital twin—The simulation aspect," in *Mechatronic Futures*, pp. 59–74. Springer, 2016. doi: 10.1007/978-3-319-32156-1_5
12. M. Dorigo, M. Birattari, M. Brambilla, "Swarm robotics and artificial swarm intelligence, in *Handbook of Artificial Intelligence*, 2014.
13. M. Hüttenrauch, A. Šošić, G. Neumann, "Deep Reinforcement Learning for Swarm Systems," *Journal of Machine Learning Research*, vol. 20, pp. 1–31, 2019.
14. M. Colledanchise, P. Ögren, *Behavior Trees in Robotics and AI: An Introduction.* CRC Press, 2018. doi: https://doi.org/10.1201/9780429489105
15. M. Dorigo, V. Maniezzo, A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Evolutionary Computation, 1(1)*, pp. 53–66, 1996. https://doi.org/10.1109/4235.585892
16. R.C. Eberhart, J. Kennedy, "A new optimizer using particle swarm theory," *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43. IEEE, 1995. doi: https://doi.org/10.1109/MHS.1995.494215
17. W. Li, H. Dong, H. Gao, "Leader-follower formation control of multi-UAVs with collision avoidance," *IEEE Transactions on Industrial Electronics*, 64(4), pp. 3184–3196, 2017. doi: https://doi.org/10.1109/TIE.2016.2638818
18. A. Marzinotto, M. Colledanchise, C. Smith, P. Ögren, "Towards a unified behavior trees framework for robot control," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5420–5427. doi: https://doi.org/10.1109/ICRA.2014.6907656
19. R. Olfati-Saber, J.A. Fax, R.M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, 95(1), pp. 215–233, 2007. doi: https://doi.org/10.1109/JPROC.2006.887293
20. V. Parunak, M. Purcell, R. O'Connell, "Digital Pheromones for Autonomous Coordination of Swarming UAV's," in *1st UAV Conference, 2002*. doi: 10.2514/6.2002-3446

21. C.W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH ‘87)*, pp. 25–34. doi: https://doi.org/10.1145/37401.37406
22. O. Vinyals et al., “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” *Nature*, 575(7782), pp. 350–354, 2019. doi: https://doi.org/10.1038/s41586-019-1724-z
23. M. Zgurovsky, Y. Zaychenko, *The Fundamentals of Computational Intelligence: System Approach*. Springer, 2016. doi: https://doi.org/10.1007/978-3-319-35162-9

Received 24.08.2025

INFORMATION ON THE ARTICLE

Michael Z. Zgurovsky, ORCID: 0000-0001-5896-7466, Educational and Scientific Complex “Institute for Applied System Analysis” of the National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Ukraine, e-mail: zgurovsm@hotmail.com

Yuriy P. Zaychenko, ORCID: 0000-0001-9662-3269, Educational and Research Institute for Applied System Analysis of the National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Ukraine, e-mail: zaychenkoyuri@ukr.net

Andrii M. Tytarenko, ORCID: 0000-0002-8265-642X, Educational and Research Institute for Applied System Analysis of the National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Ukraine, e-mail: titarenkoan@gmail.com

Oleksii V. Kuzmenko, ORCID: 0000-0003-1581-6224, Educational and Research Institute for Applied System Analysis of the National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, Ukraine, e-mail: oleksii.kuzmenko@ukr.net

МЕТОДИ РОЄВОГО ШТУЧНОГО ІНТЕЛЕКТУ В ЗАВДАННЯХ АВТОНОМНОЇ НАВІГАЦІЇ БПЛА / М.З. Згуровський, Ю.П. Зайченко, А.М. Титаренко, О.В. Кузьменко

Анотація. Подано порівняльний аналіз дев’яти методів ройового інтелекту (РІ) з точки зору їхньої придатності для бортових платформ ШІ в автономних роях безпілотних літальних апаратів (БПЛА). Визначено набір ключових критеріїв, включаючи обчислювальну складність, масштабованість, затримку, стійкість до втрати агентів та адаптивність. Децентралізовані дерева поведінки (ДП) визначені як найбільш збалансований підхід для реактивного рівня поведінки, тоді як глобальний метод оптимізації рою GBestPSO виявляється ефективним для високорівневого планування. Запропоновано гібридну двошарову когнітивну архітектуру, яка інтегрує ДП та GBestPSO, із функціональним розділенням між шарами та зв’язком на основі протоколів DDS/RTPS. Архітектура демонструє високу автономність, відмовостійкість, модульність та придатність для вбудованих систем реального часу, що працюють у динамічних або змагальних середовищах. Результати частково підтримано Національним фондом досліджень України, грант № 2025.06/0022 «Платформа штучного інтелекту з когнітивними сервісами для скоординованої автономної навігації розподілених систем, що складаються з великої кількості об’єктів».

Ключові слова: ройовий інтелект, БПЛА, автономна навігація, дерева поведінки, GBestPSO, ROS 2, DDS, когнітивна архітектура.