Self-organization of a group of mini-UAVs using the swarm intelligence

R. V. Holuboshchenko[1]

[1]Kharkiv National University of Radioelectronics, Kharkiv, Ukraine

ruslan.holuboshchenko@gmail.com

In the 21st century, unmanned aerial vehicles (UAVs) are increasingly used on the battlefield for both reconnaissance and combat operations. This technology has gained particular prominence during the Russo-Ukrainian war. Against this backdrop, discussions are intensifying around the development and deployment of fully autonomous drone swarms.

The objective of this research was to develop a basic mathematical algorithm suitable for implementation in an autonomous drone swarm tasked with reconnaissance of enemy fortifications and subsequent target engagement. The algorithm is designed to enable decentralized coordination and adaptive decision-making within the swarm under dynamic battlefield conditions.

First of all, let us assume that at any moment all agents have complete information about the map of the area under investigation and, with sufficient accuracy, about the location of enemy personnel. This will allow us to approximate the terrain by representing it as a graph and assign each node n a mark $v_n \geq 0$, which indicates the number of targets in the area approximated by that node. Additionally, we assume that the destruction of a target by a drone is a guaranteed event (i.e., each target requires exactly one drone to neutralize it).

Under these conditions, formally the problem can be stated as follows. Start with an algorithm:

Algorithm 1. *Step 1. Agent $k \in \mathcal{K}$ is located at node n. If $v_n > 0$, update $v_n = v_n - 1$, and the iteration process for this agent stops. Otherwise, proceed to step 2.*

Step 2. If $v_n = 0$, form a set $\mathcal{W}$ containing the neighboring nodes, excluding the one from which the agent arrived. Then, the agent moves based on the following principles:

(a) If $\mathcal{W}$ is empty (i.e., the agent reaches a dead end), the agent backtracks.

(b) If $\mathcal{W}$ contains a single node, the agent moves to that node.

(c) If $\mathcal{W}$ contains multiple nodes, the agent selects one based on a predefined rule $\mathcal{R}$.

Step 3. Return to step 1.

Thus, each agent k builds a certain trajectory (a sequence of nodes) of length d_k, continuing until it either completes its task or the condition $v_n = 0$ is met for all nodes. Clearly, we want an algorithm that completes the task as quickly as possible, meaning that the total distance traveled should be minimized:

$$D(\vec{d}) = \sum_{k=1}^{K} d_k \ \longrightarrow \ min, \tag{1}$$

where $K = \mathrm{card}(\mathcal{K})$ represents the total number of agents

Remark 1. Satisfying the condition $v_n = 0$ for each node does not necessarily lead to the termination of all agents' operations according to step 1 of the described algorithm if $\sum_{n=1}^{N} v_n < K$.

The rule (or strategy) $\mathcal{R}$, developed in this work and aimed at minimizing the objective function, is based on the fundamental version of the Ant Colony Optimization (ACO) algorithm [1]. Two major modifications were introduced to the original algorithm.

First, transition desire formula was changed. Since a higher pheromone level on a path implies a greater likelihood that the path is clear, the original ratio

$$W_{i,j} = \frac{\varphi_{i,j}^{\alpha}}{d_{i,j}^{\beta}}$$

was replaced with

$$W_{i,j} = \frac{1}{\varphi_{i,j}^{\alpha} d_{i,j}^{\beta}}. \tag{2}$$

Second, formula (2) does not take into account the configuration of targets on the graph, which is unrealistic – in practice, agents are expected to communicate and share information about target locations. A so-called psi-function is designed to correct this omission:

$$\psi_j = \psi(\vec{v}; \vec{d}_j) = v_j + \sum_{\substack{n=1 \\ n \neq j}}^{N} v_n \cdot \left(1 - ||d_{j,n}||\right)^4,$$

where $||d_{j,n}||$ stands for normalized distances.

Accordingly, the final form of the transition desire formula is given by:

$$W_{i,j} = \frac{\psi_j^{\omega}}{\varphi_{i,j}^{\alpha} d_{i,j}^{\beta}},$$

where ω is a real-valued parameter that controls the extent to which agents rely on the target-related signal ψ_j.

The proposed method was compared against random walk, defined as a baseline rule $\mathcal{R}$ in which nodes at step 2(c) of Algorithm 1 are selected uniformly at random [2].

The estimated mean, standard deviation, and skewness of the random walk after 200 simulation runs are:

$$\bar{X} = 23.164, \qquad \bar{\sigma} = 9.756, \qquad \bar{As} = 0.657.$$

In addition, the minimum and maximum values of the objective function observed during the simulations are:

$$L_{min} = 5.955, \qquad L_{max} = 51.242.$$

For comparison, the results obtained using the proposed method with $\omega = 2$ are as follows:

$$\bar{X} = 8.535, \quad \bar{\sigma} = 3.422, \quad \bar{As} = 1.464,$$

$$L_{min} = 3.848, \qquad L_{max} = 22.349.$$

These results demonstrate a significant improvement in average performance, reduced variability, and tighter bounds on the objective function, albeit with increased positive skewness.

This research was conducted as part of my Bachelor's thesis. I am grateful to my supervisors for providing me with a solid education, and to my mother for taking care of my basic needs so I could be fully devoted to my studies.

1. Dorigo M. Optimization, learning and natural algorithms. — Milano: Politecnico di Milano, 1992, PhD thesis, 90 p.
2. Holuboshchenko M. Study of the Swarm Intelligence Method for Multi-Parameter Problems. — Kharkiv: NURE, 2024, Bachelor's thesis, 71 p.