Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2024 SCPE. Volume 25, Issues 5, pp. 4428–4441, DOI 10.12694/scpe.v25i5.3541

# A NEW MULTI-ROBOTS SEARCH AND RESCUE STRATEGY BASED ON PENGUIN OPTIMIZATION ALGORITHM

OUARDA ZEDADRA *, AMINA ZEDADRA †, ANTONIO GUERRIERI ‡, HAMID SERIDI §, AND DOUAA GHELIS ¶

**Abstract.** In response to the challenging conditions that arise after natural disasters, multi-robot systems are utilized as alternatives to humans for searching and rescuing victims. Exploring unknown environments is crucial in mobile robotics, serving as a foundational stage for applications such as search and rescue, cleaning tasks, and foraging. In our study, we introduced a novel search strategy for multi-robot search and rescue operations. This strategy draws inspiration from the hunting behavior of penguins and combines the Penguin Search Optimization Algorithm with the Random Walk Algorithm to regulate the global and local search behaviors of the robots. To assess the strategy's effectiveness, we implemented it in the ARGoS multi-robot simulator and conducted a series of experiments. The results clearly demonstrate the efficiency and effectiveness of our proposed search strategy.

**Key words:** Swarm Intelligence, Swarm Robotics, Search and Rescue Problem, Penguin Search Optimization Algorithm, Random Walk Algorithm.

**1. Introduction.** Exploring unknown environments is a fundamental concern in mobile robotics, crucial for applications like cleaning, search and rescue, and foraging. Search and rescue operations play a critical role in disaster management, aiming to ensure the safety of individuals and minimize rescue time. While these operations can be challenging for human rescuers, mobile robots are utilized to navigate and explore disaster-stricken areas, locate victims, and perform various tasks that enhance the effectiveness and efficiency of rescue operations. They can access hazardous or inaccessible areas, gather information, and provide assistance to humans.

In search and rescue operations, the effectiveness of multi-robot systems lies in their capacity to cover extensive areas, gather more data, and improve operational efficiency, particularly in tasks deemed perilous, time-consuming, or beyond human capabilities. Despite these advantages, coordinating and communicating among robots present significant challenges. Effective collaboration and information sharing are crucial for comprehensive search area coverage. Designing systems that facilitate seamless coordination and communication is essential to maximize the multi-robot team's effectiveness. Additionally, the ability of these systems to operate in environments with limited communication and navigation capabilities is crucial, considering that search and rescue scenarios often occur in areas where communication infrastructure may be damaged or nonexistent. Therefore, multi-robot systems need robust communication and adaptable search methods for such challenging environments.

Swarm intelligence has been a source of inspiration for various applications of mobile robotics. For example, some animals' exploration, return, and communication strategies have shown great effectiveness, especially in mobile robot exploration. In this work, we proposed a new search and rescue strategy based on swarm intelligence algorithms to achieve the following objectives:

1. Increase the explored areas.

*LabSTIC Laboratory, Department of Computer Science, 8 May 1945 University, P.O. Box 401, Guelma, Algeria (zedadra.ouarda@univ-guelma.dz).

†LabSTIC Laboratory, Department of Computer Science, 8 May 1945 University, P.O. Box 401, Guelma, Algeria (zedadra.amina@univ-guelma.dz)

‡National Research Council of Italy, Institute for High Performance Computing and Networking (ICAR), Via P. Bucci 8/9C, 87036 Rende, Italy (antonio.guerrieri@icar.cnr.it).

§LabSTIC Laboratory, Department of Computer Science, 8 May 1945 University, P.O. Box 401, Guelma, Algeria (seridi.hamid@univ-guelma.dz)

¶Department of Computer Science, 8 May 1945 University, P.O. Box 401, Guelma, Algeria (douaaghelis41@gmail.com)

2. Disperse robots widely in their environment to increase the number of found victims.
3. Reduce search time.
4. Decrease the visits to already explored areas.

The majority of research in the search and rescue field predominantly relies on old SI algorithms (mature algorithms such as Ant Colony Optimization, i.e., ACO, Particle Swarm Optimization, i.e., PSO, Artificial Bee Colony, i.e., ABC). The Penguin Optimization Algorithm (POA) is an optimization technique inspired by the cooperative hunting strategies of penguins, specifically their ability to work together to locate and capture fish in harsh and dynamic environments. POA simulates this behavior by modeling the search process as a collaborative effort among multiple agents, each representing a penguin, to explore the solution space and converge on optimal solutions. To the best of our knowledge, the Penguin Search Optimization Algorithm (PeSOA) was used to optimize mathematical benchmark functions and has not yet been used in multi-robot-related problems. That is why we adapted and used the PeSOA in search and rescue problems. The POA is particularly well-suited for the problem of multi-robot search and rescue for several reasons:

- *Cooperative Behavior:* Similar to how penguins coordinate their movements to find food efficiently, POA enables multiple robots to collaborate, share information, and optimize their search patterns in a coordinated manner, enhancing the overall efficiency of search and rescue operations.
- *Adaptability:* The dynamic nature of the POA allows it to adapt to changing environmental conditions and obstacles, which is critical in unpredictable search and rescue scenarios where obstacles and conditions can vary widely.
- *Robustness:* The algorithm's robustness in dealing with complex and dynamic environments ensures that the robots can effectively navigate and perform their tasks despite uncertainties.

The paper is organized as follows: in Section 2, we introduce the Penguin Optimization Algorithm and summarize some recent related works. In Section 3, we present the finite state machine and the pseudo-code of the proposed algorithm. In Section 4, we present and discuss the obtained results. Finally, in Section 5, we conclude the work and present some perspectives.

**2. State of the art.** In this Section, we first present a description of the Penguin Optimization Algorithm (PeSOA). Then, we summarize some relevant and recent works related to the search and rescue problem and present a qualitative comparison of the reviewed works based on some relevant characteristics.

**2.1. Description of the Penguin Optimization Algorithm (PeSOA).** The Penguin Optimization Algorithm is a meta-heuristic algorithm inspired by the collaborative hunting strategies of penguins. This bio-inspired algorithm mimics the way penguins hunt for fish in groups, optimizing their energy expenditure and maximizing their food intake.

*Hunting Strategy of Penguins.* Penguins hunt in groups and use a strategy that involves synchronized dives and communication to locate and capture fish. Each penguin searches for food individually and then communicates its findings to the group. This collaboration allows the group to identify the areas with the highest concentration of food and optimize their foraging efforts. The key behaviors modeled in PeSOA include [1]:

- Group Hunting: Penguins hunt in groups, coordinating their efforts to maximize efficiency.
- Communication: Penguins communicate their positions and the amount of food found to the group, enabling collective decision-making.
- Synchronized Dives: Penguins synchronize their dives to systematically search different depths and areas.

*Algorithm Structure.* PeSOA simulates these behaviors to solve optimization problems by following these steps [1]:

1. Initialization: Generate an initial population of solutions (penguins) grouped into several groups.
2. Random Search: Each penguin performs a random search within a defined range (hole and level) until its oxygen reserves are depleted.
3. Communication: After the search, penguins return to the surface and share their findings (location and amount of food) with their group.
4. Update Positions: Penguins update their positions and probabilities of finding food in different locations based on the shared information.

5. Iteration: The process repeats for a number of generations, with the group collectively moving towards the areas with the highest food concentration (optimal solutions).

*Key Components.* The algorithm is composed of the components below [2]:

- Population Groups: Penguins are divided into groups, and each group searches specific areas.
- Intra-Group Communication: Penguins within the same group share their findings to refine their search strategy.
- Inter-Group Communication: Groups with less success may follow the more successful groups to improve their chances of finding food.
- Probabilistic Search: The search process is guided by the probability of food presence, which is updated based on the findings.

*Mathematical Formulation.* The position of each penguin is updated using the following equation:

$$D_{new} = D_{last} + rand() \times |X_{localbest} - X_{last}| \tag{2.1}$$

where $D\_new$ is the new position, $D\_last$ is the last position, $rand()$ is a random number for distribution, $X\_localbest$ is the best local solution, $X\_last$ is the last solution.

PeSOA has been validated against several benchmark functions, such as the Rastrigin and Schwefel functions, and compared with other optimization algorithms like Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). The results showed that PeSOA is robust and efficient, particularly in avoiding local optima and finding global solutions.

**2.2. Related works.** In the literature, Firthous and Kumar [10] introduced a new search and rescue algorithm called *Artificial Bee Colony with Modified Evolutionary Programming (ABCMEP)*, combining ABC and Modified Evolutionary Programming (MEP). ABC is used for obstacle avoidance, while MEP optimizes the path for efficiency. Simulated in MATLAB, the algorithm was tested in various environments, comparing it with ABC and ABCEP (Artificial Bee Colony with Evolutionary Programming). The ABCMEP algorithm demonstrated superior performance, providing efficient target localization in unknown areas by avoiding obstacles. The study presents the best cost values for each algorithm in different environments, highlighting the effectiveness of ABCMEP.

Garg et al. [7] introduce the Velocity-inspired Robotic Fruit Fly Algorithm (VRFA) as a search strategy for efficient victim search and real-time assessment in search and rescue operations. VRFA combines the Fruit Fly Optimization algorithm (FOA) for target search and tracking and the PSO algorithm for updating positions and velocities. The independent robot in the system performs activities such as local data processing, data sharing, movement plan formulation, and timeline generation. The study compares VRFA with other techniques using centralized and decentralized cooperation strategies for both static and dynamic targets. Results indicate that VRFA outperforms other algorithms, particularly in decentralized cooperation, demonstrating reduced search and rescue time and superior performance in dynamic scenarios.

Huang et al. [9] propose a novel algorithm for task allocation in a multi-robot cooperative rescue system, integrating the Ant Colony Contract Net Protocol (ACCNP). The algorithm uses the ACO algorithm to determine the initial allocation results and the CNP algorithm for dynamic adjustments in changing environments. Notable features include parallel computation, time efficiency, and adaptability to environmental changes. Simulated experiments in a fire rescue scenario reveal the CNP algorithm's superiority over ACO in terms of computation time, average task completion cost, and average task completion time. The results emphasize the effectiveness of CNP in solving multi-robot task assignment challenges within dynamic environments.

In their paper [12], the authors introduce a novel strategy called Distributed Particle Swarm Optimization (DPSO) for multi-robot exploration in search and rescue operations. The focus is on guiding a robot swarm to navigate the search space, avoid obstacles, and locate victims. Key concepts include robots avoiding static and dynamic obstacles, victims scattered randomly, robots sensing local environments, and sharing positions. DPSO addresses the drawbacks of classical PSO by incorporating an artificial potential function to attract forces toward unknown areas and victims, along with a repulsion forces function for collision avoidance, divided into intra and inter-repulsion forces. Experiments were conducted using Python and VRep for multi-agent model implementation and environmental simulation. Four experiments demonstrated that the DPSO algorithm

enables robots to escape local minima, find alternative paths, and navigate through disaster scenarios effectively, leading to optimal solutions.

Dah-Achinanon et al. [5] present a search and rescue algorithm utilizing ad-hoc networks with sporadic connectivity. The algorithm aims to bridge the gap between theoretical concepts and practical applications, emphasizing autonomy, decentralization, and effective research strategy. It facilitates communication among swarm members to share information about target findings and locations, enabling a base station to coordinate rescue efforts. The search method employs belief space exploration, incorporating key information from authorities, such as the last known locations of targets. The algorithm's principles include individual drones conducting searches based on available belief information, updating and distributing belief maps to prevent redundant searches, and using virtual stigmergy for belief map distribution. Validation of the algorithm involves tests in the ARGOS simulator platform and real-world experiments, assessing the time required for target discovery and relay chain establishment. Results from simulations and real-world tests confirm the feasibility and effectiveness of the proposed approach, demonstrating superiority over a random walk strategy.

In [13], authors describe an adverse search and rescue Multi-Agent Reinforcement Learning (MARL) approach to learning efficient coordination and collaboration strategies to achieve search and rescue mission objectives in the presence of adverse search and rescue communications. A Centralized Training approach with Decentralized Execution is used in this training approach. The aim is to coordinate the search to avoid visiting the same location multiple times and reduce the overall target exploration time by cooperative agents in the presence of adverse search and rescue. To investigate how adverse search and rescue interference can derail the performance of cooperative agents in a search and rescue mission and to what extent the proposed training algorithm can mitigate adverse search and rescue actions, the authors conducted four case studies to evaluate the proposed model. According to each case's results, the agents could locate the targets more successfully in the case with the modified reward structure.

The difficulties of search and rescue operations when the space is difficult to cover by human rescuers and ground robots have led to the emergence of a new technology for use, namely Unmanned Aerial Vehicles (UAVs). Authors in [14] proposed a Self-Organizing Mesh Network that is optimized for area coverage by maximizing the search area and maintaining wireless communication. This new system lays the groundwork for the behavior of more than two unmanned aerial vehicles. Whenever a UAV is connected to more than one other UAV, it will attempt to remain within the range of all of them. Additionally, this system prevents UAVs belonging to the same swarm from colliding with one another. Rather than having a leading coordinator UAV, all of the UAVs adjust based on each other. The authors proposed a mesh reliance architecture, which works by assigning the swarm a connectivity matrix, with each UAV connection being a rep and a swarm controller based on a Genetic Algorithm (GA) and Neural Network (NN). Based on the simulation results, the proposed methods are valid for organizing swarms. However, they are not as effective for swarm travel, and adding each drone did increase computation time. GA runs very slowly, making it a poor choice in emergencies.

In order to solve the problems of full-coverage path planning in search and rescue operations, which strive to completely cover the area of interest in a limited time and avoid obstacles autonomously in unknown areas, the authors in [16] proposed a new Neural Network Algorithm through the ABC algorithm. The Neural Network takes as input information about obstacles and coverage in the five directions and gives as output the speed of the left and right wheels. In the initial situation, the parameters are randomly generated, giving the path the neural network planned a lower score. According to the advantages of the ABC algorithm, such as the increased probability of finding the optimal solution and the robustness of the system, the authors used this method to optimize the parameters of the Neural Network and improve the training efficiency and effects of the entire system. To analyze the algorithm's performance, the authors define an evaluation function, which is divided into three parts: the coverage rate, the path repetition rate, and the failure rate. Based on experimental results, the ABC method, combined with a Neural Network path planning algorithm, can effectively control rescue robots to plan complete coverage paths and can be migrated to various environments with high robustness.

Naval operations such as marine search and rescue operations require a higher performance and efficient control to rescue survivors. The authors [15] presented a new method for visual navigation and Unmanned Surface Vehicle (USV) control named Conventional Convolutional Neural Network Spatial Softmax (CNN-SS). The authors divided their work into two main parts, summarized as follows:

- Deep learning-based visual navigation architecture for USV and floating object positioning in USV-UAV operations. This approach improves visual positioning accuracy and computational efficiency by introducing a soft-max spatial layer and a two-stage structure.
- Reinforcement Learning (RL)based USV control strategy approaches and encircles a floating object. The trained control policy can improve the control system's performance under wave disturbances by adding observable wave features to the state space.

The proposed visual navigation architecture consists of two stages: the first consists of estimating the position of the USV and the floating object, and the second stage consists of estimating the heading angle of the USV. This two-stage architecture is built based on CNN and the network structure. A USV controller based on RL has been developed to avoid collisions with floating objects. The basic principle of RL is to maximize the accumulated reward at every step by iteratively optimizing a policy. The authors conducted several experiments to evaluate the performance of the visual navigation model and the USV controller model. They evaluated four network models (CNN-SS, Conventional Convolutional Neural Network (CNN), Fully Convolutional Network (FCN), YOLOv5) for the first proposed algorithm and designed three tasks to evaluate the second algorithm. Therefore, the proposed visual navigation architecture can provide high-accuracy position, velocity, and handing angle estimations under most weather conditions. However, this latter still has some limitations in heavy foggy weather. Finally, after the simulation, the authors demonstrated the effectiveness and superiority of the proposed algorithms.

Authors in [11] developed a new algorithm called the Ant Search Path with Visibility algorithm (ASPV) for the NP-hard optimal search path problem with visibility. This algorithm draws inspiration from ACO principles, which define 96 variants based on four key components of traditional ACO: pheromone initialization, pheromone update, restart, and boosting. The authors conducted several experiments to identify the best variant of the proposed ACO algorithm variants. In these experiments, three phases were conducted:

- Configuration phase, which determines the optimal parameter pairs for each variant.
- multi-run evaluation phase allows for determining how the performance of an ACO varies between runs on a given instance.
- The across-instance evaluation phase provides a better understanding of an ACO's performance in a variety of instances.

The authors compared the performance of the best ASPV algorithm variants with that obtained through a Mixed-Integer Linear Program (MILP) and with a simple greedy heuristic. Results indicated that the proposed algorithm produced search paths with higher success probabilities in a shorter period.

In the study by AI et al. [3], a novel autonomous coverage planning model for maritime search and rescue operations is introduced, leveraging reinforcement learning. The key contributions include (1) complex environment modeling, and (2) introducing a reinforcement learning algorithm with a multi-objective reward function. This function considers avoiding obstacles, preventing repeated paths, and favoring high-probability areas. The algorithm effectively addresses issues like sparse rewards, sub-goal conflicts, and reduces overlapping paths during learning, (3) developing an action selection policy that balances exploitation and exploration to determine the global-optimal solution. Experiments in a simulation area demonstrate the model's validity. Comparisons with various trajectory planning algorithms, including Boustrophedon Motion (BM), Boustrophedon Cellular Decomposition (BCD), and Boustrophedon Motions with the A* search (BA*), show that the proposed algorithm generates search paths covering the entire search and rescue area safely, with low repetition rates and prioritization of high-probability areas for searching.

Authors in [6] proposed a new extension of the A* algorithm for the 3D search and rescue environment. This algorithm is based on the A* and Task Allocation algorithms to obtain a faster and more efficient path-planning method. The objectives achieved by the author are summarized below:

- General 2D and 3D trajectory planning of UAVs.
- Granting UAVs the ability to avoid obstacles when performing tasks.
- Identifying the shortest path for all UAVs in a minimum of time.

In this paper, the task allocation algorithm provides advantages to the system, such as a parallel search of the environment, allowing the drones to complete the entire task by distributing the assignments to each drone, helping to reduce the total route planning time and memory space of each drone. Moreover, the A* algorithm

is generally used for the path planning optimization problem. To evaluate the performance of the proposed algorithm, the author created 2D and 3D simulation scenarios to test the functionality of the task allocation algorithm and performed experiments in a 3D environment to verify and evaluate the performance of the improved A* algorithm compared to the classical A* algorithm. According to the results obtained from the simulation, it can be observed that the improved algorithm has better running time, speed, and efficiency than the classical algorithm and has higher efficiency in the coverage environment, and reduces the amount of storage required.

Because of the increase in the number of casualties in the maritime sector, maritime authorities and operation centers are trying to develop a quick search for survivors at sea using different technologies such as USV, Unmanned Underwater Vehicle (UUV), and UAV. [4] introduced a new contribution by using a UAV swarm. This latter was divided into a two-phase method to solve the (CPP) problem. The first phase presents a methodology for transforming the search area into a graph consisting of vertices and edges using grid-based area decomposition. Hence, the proposed method takes into account the angle at which the area is decomposed to minimize the size of the coverage area. The second phase focuses on determining the optimal path for multiple UAVs based on the results of the first phase to reduce the completion time. The authors developed a Mixed-Integer Linear Programming (MILP) model that minimizes the time required to cover the search area. As a constraint, the region (position of nodes, the angle between nodes, etc.) and the dynamics of the robot covering the nodes are considered. The researchers in this paper observed that the proposed MILP model was time-consuming for large-scale real-world problems. To solve this problem, a new algorithm called Randomness Search Heuristic (RSH) was developed. This approach consists of three phases in each iteration:

1. Random construction: constructs the coverage path of UAVs by sequentially selecting the adjacent nodes.
2. Repair: repairs the generated roads if all nodes are uncovered to recover the feasibility.
3. Local search: improves the constructed path for each UAV to obtain high-quality solutions that are close to the optimal road.

This paper considered numerical experiments to validate the efficiency and effectiveness of the proposed approach and real-world experiments with two UAVs to verify the validity of the proposed algorithm in the real world. According to the results of the numerical experiments, the RSH algorithm can produce near-optimal solutions in a much shorter computational time than a commercial solver. In contrast to the real-world experiments, the mission execution time was longer due to the influence of wind or network communication uncertainty.

After summarizing the different related works, we conclude some important points:

1. Based on Table 2.1, we can group the research according to the scenarios used in the different search and rescue operations (based-ground search and rescue, maritime search and rescue, and aerial search and rescue).
2. Some works used a single sink with a fixed position, the others did not specify the number of sinks and their positions.
3. Most of the works have used static targets with a random position.
4. The majority of the works used multiple robots with a random position.
5. All research done in the based-ground search and rescue focuses on homogeneous robots and is based on decentralized control.
6. Most research focuses on cooperation between robots using direct communication.
7. The majority of studies take into account the lack of redundancy in space exploration, and all of these studies use different search methods.
8. In the search and rescue based-ground, all research experiments were conducted by simulation using different simulators.
9. As for all the works, the use of swarm intelligence-based algorithms is very weak and needs to be investigated since these algorithms provide scalable, flexible, and robust solutions to systems like search and rescue ones.

**3. Proposed Algorithm: Modified Penguin Search Optimization Algorithm (MPeSOA).** In this Section, we present the proposed search and rescue algorithm named: The Modified Penguin Search Optimization Algorithm (MPeSOA). During its life cycle, the robot can be in one of the four behaviors: **Global**

Table 2.1: Qualitative comparison of the summarized related works.

| | | | [10] | [7] | [8] | [9] | [16] | [12] | [13] | [14] | [5] | [15] | [11] | [3] | [6] | [4] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Environment | Complexity | with obstacles | X | X | | | X | X | X | | X | | | X | | |
| | | Free of obstacles | | | X | X | | | | X | | X | X | | X | X |
| Object | Distribution | Random | | X | | | | X | X | | X | X | X | | X | |
| | | fixed | X | | | X | | | | | | | | | | |
| | | clustered | | | X | | | | | | | | | | | |
| | Nature | Static | X | X | X | X | | X | X | | X | X | | | X | |
| | | Dynamic | | X | | | | | | | | X | X | | | |
| Robot | Position | fixed | X | | | X | X | | | | | | | X | | X |
| | | random | | X | X | | | X | X | X | X | X | X | | X | |
| | Homogeneity | Yes | X | X | X | X | X | X | X | X | X | | X | X | X | X |
| | | No | | | | | | | | | | X | | | | |
| | Number | single | X | | | | X | | | | | | | X | | |
| | | multiple | | X | X | X | | X | X | X | X | X | X | | X | X |
| Strategy | control | centralized | | X | X | | | | | | | | | | | |
| | | decentralized | X | X | | X | | X | X | X | X | X | X | X | X | X |
| | cooperation | Yes | | X | X | X | | X | X | X | X | X | X | | X | |
| | | No | X | | | | X | | | | | | | X | | X |
| | communication | direct | | X | X | X | | X | | X | X | X | X | | X | |
| | | indirect | | | | | | | X | | | | | | | |
| | exploration redundancy | Yes | | X | | | X | | | | | | X | | | |
| | | No | X | | X | X | | X | X | X | X | | X | X | X | X |
| | exploration type | ABCEMP | X | | | | | | | | | | | | | |
| | | VRFA | | X | | | | | | | | | | | | |
| | | CNP | | | | X | | | | | | | | | | |
| | | MARL | | | | | | | X | | | | | | | |
| | | ABC | | | | | X | | | | | | | | | |
| | | ANN | | | | | X | | | | | | | | | |
| | | belief based search | | | | | | | | | X | | | | | |
| | | GA | | | | | | | | X | | | | | | |
| | | leader follower | | | X | | | | | | | | | | | |
| | | D-PSO | | | | | | X | | | | | | | | |
| | | DL | | | | | | | | | | X | | | | |
| | | RL | | | | | | | | | | X | | X | | |
| | | ASPV | | | | | | | | | | X | | | | |
| | | enhanced $A^*$ | | | | | | | | | | | | | X | |
| | | RSH | | | | | | | | | | | | | | X |
| Simulations | Real world | Real experiments | | | | | | | | | X | | | | | X |
| | Simulator | ArGOS | | | | | | | | | X | | | | | |
| | | Netlogo | | X | | | | | | | | | | | | |
| | | Python | | | X | X | | | X | | | X | | X | | X |
| | | Pybullet | | | X | | | X | | | | | | | | |
| | | VRep | | | | | | X | | | | | | | | |
| | | MATLAB | X | | | | | | | | | | | | X | |
| | | $C^{++}$ | | | | | | | | | | | X | | | |

**Search**, **Local Search**, **Migration**, and **Obstacle Avoidance**. The robots start all from the central depot and change their behaviors according to input data gathered by their sensors. The state machine representing the behavior of the robots is given by Figure 3.1.

We also present in this section a pseudo code of the **proposed MPeSOA algorithm** (Algorithm 1), and the pseudo-codes of the four key states: **Global search** (Algorithm 2), **Local search** (Algorithm 3), **Obstacle avoidance** (Algorithm 5), and **Migration** (Algorithm 4).

Below, we explain the different behaviors of the state machine. Throughout the subsequent rules, the oxygen reserve O(t) will be treated as a constant value set to 1.

*Global search.* During this phase, each group of robots undergoes a relocation process to a new position, guided by the LocalBest solution obtained from the previous dive. This relocation is executed by employing the penguin search position equation as depicted in Equation 3.1. Individual robots may encounter obstacles or victims throughout the search process, prompting distinct decision-making. For instance, when faced with

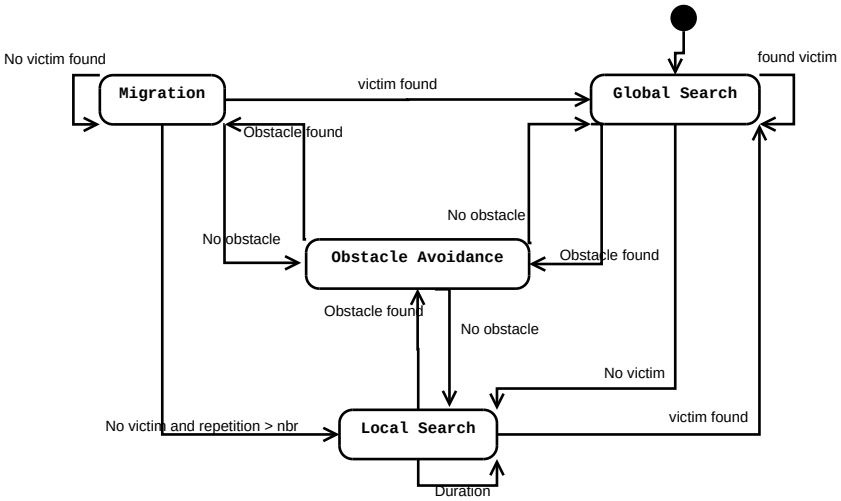



Fig. 3.1: MPeSOA's State Machine.

```
Initialize the number of groups;
Initialize the total number of robots;
Generate a random local best for each group;
Distribute randomly the robots in groups;
while (stopping criteria is not reached) do
    for each group i do
        Goto Global Search (Algorithm 2);
        Update the food abundance degree for this group using Equation 3.4;
    Update the total eating food;
    Update the global best solution;
    Update membership function values for each group by Equation 3.3;
```

**Algorithm 1:** Pseudo code of the MPeSOA Algorithm.

obstacles, a robot must avoid them using the *obstacle avoidance state*. Conversely, the robot must prioritize its rescue and subsequent safety upon discovering victims. Once the group reaches a new position, all group members must shift their state to the Local Search state. Algorithm 2 presents the pseudo-code of the Global search state.

$$x_j^i(t+1) = x_j^i(t) + rand() \times \left| x_{Localbest}^i - x_j^i(t) \right| \tag{3.1}$$

where: $x_j^i(t)$ is the position of penguin $j$ allocated to the $i^{th}$ group at $t^{th}$ instant. $x_{localbest}^i$ is the best solution found by $i^{th}$ group. $rand(a, b)$ is a random number drawing from $(0, 1)$

*Local Search.* Before starting the local search, the robots must await the arrival of all group members, ensuring the collective initiation of the search procedure employing the Random Walk Algorithm (RWA) (Equation 3.2). As in the Global search, the robots have to get around obstacles and effectively rescue victims encountered along the way. However, the key distinction lies in the search duration. In this case, the robots apply the RWA for a predefined period and subsequently repeat this process in the absence of victim discoveries, thereby providing additional opportunities to locate victims. In instances where no victims are detected, the robots must undergo migration by reverting to the Migration state. Conversely, upon successful victim detection, the group updates their local best and transitions to the Global search state. The pseudo-code of this state is

```
calculate new position using Equation 3.1;
while the group does not reach the new position do
    if ∃ obstacle then
        Goto Obstacle Avoidance (Algorithm 5);
    else
        if ∃ victim then
            get the victim;
            update the number of victims rescued;
            wait a few time before updating localbest for this group;
            if the time is expired then
                update localbest for this group;
                update the quantity of eating fish for this group;
                Goto Global Search (Algorithm 2);
if the group reached the new position then
    Goto local Search (Algorithm 3);
```

**Algorithm 2:** Pseudo code of the global search Algorithm.

represented by Algorithm 3.

$$sl_i = rand(a, b) \tag{3.2}$$

where: $sl_i$ is the step length for the $i^{th}$, and $rand(a, b)$ is a random number drawing from $(a, b)$

```
while repetition process > 0 do
    while the duration has not expired do
        while ∄ obstacle or victim or pheromone do
            Random walk using Equation 3.2;
        if (∃ obstacle) then
            Goto Obstacle Avoidance (Algorithm 5);
        else
            if (∃ victim) then
                Get the victim;
                Update the number of victims rescued;
                Update the quantity of eaten fish for this group;
                Update duration for this group;
    Update LocalBest for this group;
    if new localbest=the last localbest then
        decrease repetition process;
        if repetition process =0 then
            Goto Migration (Algorithm 4);
    else
        Goto Global Search (Algorithm 2);
```

**Algorithm 3:** Pseudo code of the Local Search Algorithm.

*Migration State.* If victims remain undetected, the group will initiate migration and merge with another group. Before the migration process, it is necessary to search and rescue to compute the probabilities of the existence of fish for all groups using Equation 3.3. However, if all computed probabilities are found to be less than 0.5, the current group will refrain from migrating and take a random localbest. Conversely, if at least one probability exceeds or equals 0.5, the group will initiate migration based on the aforementioned approach

outlined in the Migration State. The equation governing the update of the Quantity of Eating Fish is equation 3.4:

$$P_j(t+1) = \frac{QEF^i(t)}{\sum_{i=1}^{k} QEF^j(t)} \tag{3.3}$$

$$QEF^i(t) = \sum_{j=1}^{di} E_j^i(t) \tag{3.4}$$

where: $E_j^i(t)$ represents the quantity of eaten fish of the $j^{th}$ robot of $i^{th}$ group at $t^{th}$ instance.

```
Return to the best position;
calculate Pi using Equation 3.3;
if all probabilities < 0.5 then
    set a random localbest;
    Goto Global Search (Algorithm 2);
else
    divide the group into two subgroups;
    give the subgroups opposite directions;
    while ∄ obstacle or victim or pheromone do
        use the same random step for all members within the same group until they reach the hunting group;
    if ∃ obstacle then
        Goto Obstacle Avoidance (Algorithm 5);
    else
        if ∃ victim then
            get the victim;
            update the number of victims rescued;
            update localbest for this group;
            update the quantity of eating fish for this group;
            Goto Global Search (Algorithm 2);
```

**Algorithm 4:** Pseudo code of the Migration Algorithm.

*Obstacle avoidance.* Our explorer robot is equipped with 24 IR proximity sensors positioned strategically around its structure. These sensors possess detect obstacles within a proximity range of 30 cm. In accordance with the readings obtained from these sensors, if the robot identifies the presence of an obstacle, it retrieves the angle information and adjusts its movement accordingly. When encountering a negative angle, the robot executes a right turn, whereas a positive angle prompts a left turn.

```
Get readings from all proximity sensors;
Accumulate all readings and get the accumulated value and the angle;
if accumulated value > 0 then
    if accumulated angle > 0 then
        Turn left;
    else
        Turn right;
```

**Algorithm 5:** Pseudo code of the obstacle avoidance Algorithm.

Table 4.1: The proposed simulation scenarios

| **Scenario 1: the influence of robot's number** | **Scenario 2: the influence of group's number** |
|---|---|
| number of robots = 30, 40, 50, 60<br>victim distribution: clusters<br>number of groups: 6<br>number of clusters: 12<br>number of victims: 910<br>obstacle density: 10%<br>environment size: 120m X 120m | number of robots = 50<br>victim distribution: clusters<br>number of groups: 3, 5, 6, 8<br>number of clusters: 12<br>number of victims: 910<br>obstacle density: 10%<br>environment size: 120m X 120m |
| **Scenario 3: the influence of cluster's number** | **Scenario 4: the influence of environment size** |
| number of robots = 50<br>victim distribution: clusters<br>number of groups: 6<br>number of clusters: 2, 4, 8, 16<br>number of victims: 1000<br>obstacle density: 10%<br>environment size: 120m X 120m | number of robots = 50<br>victim distribution: clusters<br>number of groups: 6<br>number of clusters: 12<br>number of victims: 910<br>obstacle density: 10%<br>environment size: 80m X 80m, 100m X 100m, 150m X 150m, 190m X 190m |

**4. Results and Discussions.** In order to validate the proposed algorithm (PeSOA), we implemented it by using the ARGoS mobile robotics simulator. Finding the right parameters for the algorithm will certainly increase its efficiency. We conducted various simulations to test the influence of the different parameters on the performance of the proposed algorithm. The considered parameters comprehend (i) the number of robots, (ii) the number of groups, (iii) the number of clusters, and (iv) the environment size. Moreover, we used as a performance criterion the number of collected victims in a fixed time (20 minutes).

**4.1. Simulation scenarios.** We provide in this section an overview of four simulation scenarios. We test the influence of robot numbers, group numbers, cluster numbers, and environment sizes on the performance of the proposed strategy by using these scenarios. Table 4.1 shows the different proposed scenarios:

**4.2. Results and discussions.** We introduce and analyze in this section the results obtained by applying this algorithm using scenarios presented in Table 4.1. It is important to note that the reported results represent the average outcome from five simulations.

*Results of scenario 1.* As the number of robots increases, the total number of found victims also increases. Increasing the number of robots positively impacts search and rescue operations. However, the rate of increase in the number of found victims is non-linear with the increase in the number of robots. The rate of increase diminishes as the number of robots increases. When the number of robots increases from 30 to 40 robots, the number of found victims is increased by 27, while going from 50 to 60 robots, it is increased by 156. After 60 robots, the performances did not significantly improve. Table 4.2 and Figure 4.1.a show the results obtained in this scenario.

Table 4.2: Results of scenario 1

| **Number of robots** | 30 | 40 | 50 | 60 |
|---|---|---|---|---|
| **MPeSOA** | 40% | 43% | 48% | 65% |

*Results of scenario 2.* With a smaller number of groups, there might be fewer opportunities for efficient coverage of the search area. Increasing the number of groups leads to a notable improvement in the number of found victims. This indicates that the algorithm benefits from finer granularity in dividing the search area into smaller sections, enabling more thorough exploration and detection of victims. However, the rate of improvement starts to diminish compared to the previous increase from 3 to 5 groups. The results indicate that

there's an optimal number of groups for maximizing the effectiveness of the search and rescue operation. Also, increasing the number of groups allows for better coverage of the search area but also introduces challenges in coordination among the groups. Table 4.3 and Figure 4.1.b show the results obtained in this scenario.

Table 4.3: Results of scenario 2

| **Number of groups** | 3 | 5 | 6 | 8 |
|---|---|---|---|---|
| **MPeSOA** | 38% | 48% | 64% | 72% |

*Results of scenario 3.* The algorithm achieves the maximum possible number of found victims when all clusters are thoroughly searched. This suggests that having a smaller number of clusters simplifies the search process and maximizes coverage within each cluster. Further increasing the number of clusters leads to a significant decrease in the total number of found victims. With eight clusters, the search area becomes more fragmented, making it challenging for the robots to cover each cluster thoroughly. Additionally, coordinating movements and managing resources across more clusters might introduce inefficiencies or delays in the search and rescue process. Table 4.4 and Figure 4.1.c show the results obtained in this scenario.

Table 4.4: Results of scenario 3

| **Number of clusters** | 2 | 4 | 8 | 16 |
|---|---|---|---|---|
| **MPeSOA** | 100% | 75% | 60% | 57% |

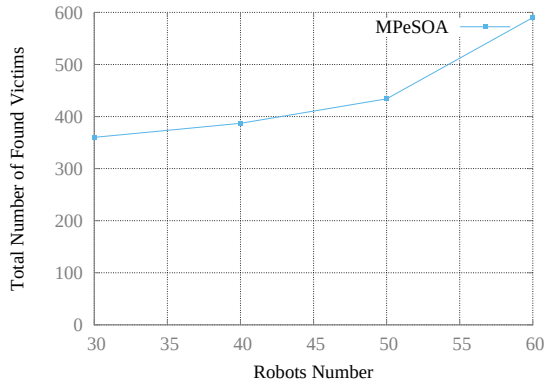
*Results of scenario 4.* The results indicate that the environment size has a significant impact on the algorithm's performance in terms of finding victims. As the environment size increases, the total number of found victims decreases consistently. When the environment size increases from 80x80 to 100x100, there's a noticeable decrease in the number of found victims. This reduction suggests that expanding the search space slightly beyond a certain threshold can lead to decreased search efficiency. The trend continues as the environment size increases to 150x150 and 190x190, with a corresponding decrease in the number of found victims. The rate of decrease appears to accelerate as the environment size grows larger. This indicates diminishing returns in search efficiency with further expansion of the search area. Table 4.4 and Figure 4.1.d show the results obtained in this scenario.

Table 4.5: Results of scenario 4

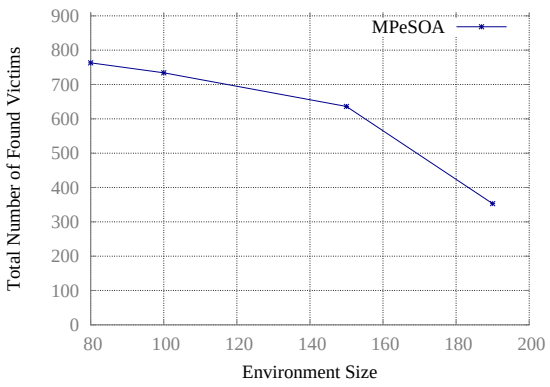
| **Environment size** | 80X80 | 100X100 | 150X150 | 190X190 |
|---|---|---|---|---|
| **MPeSOA** | 84% | 81% | 70% | 39% |

**4.3. Recommendations for parameters settings.** Based on the analysis of the results from the experiments detailed in the document, here are some extended recommendations on how to set the studied parameters to enhance the effectiveness of the search and rescue strategy using the Penguin Optimization Algorithm (PeSOA):

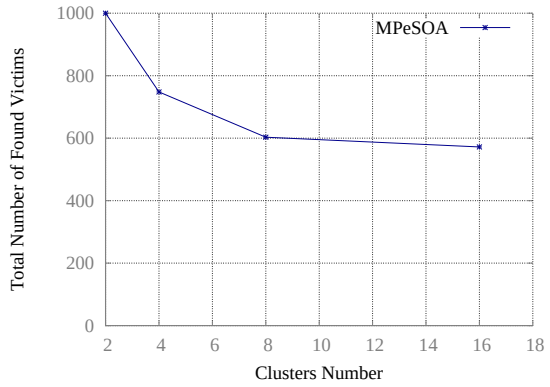
1. *Number of Robots:* Deploy a fleet of around 50-60 robots for optimal performance without unnecessary redundancy. This range maximizes the number of found victims while maintaining efficiency.
2. *Number of Groups:* Use 6-8 groups to balance the coverage area and coordination complexity. This setting ensures thorough exploration and effective victim detection while managing inter-group communication efficiently.
3. *Number of Clusters:* Keep the number of clusters low (2-4 clusters). This approach simplifies the search process, allows more thorough coverage of each cluster, and enhances the efficiency of the overall search and rescue operation.



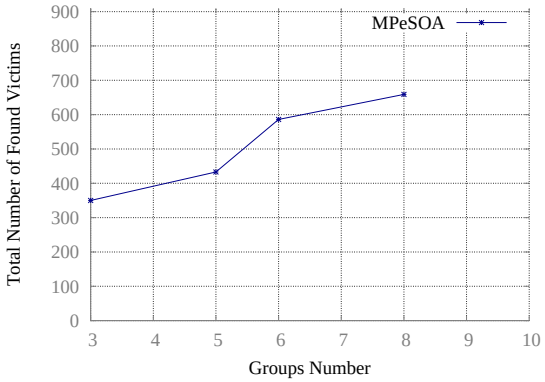

(a) Results of scenario 1




(b) Results of scenario 2




(c) Results of scenario 3




(d) Results of scenario 4

Fig. 4.1: Simulation results

4. *Environment Size:* For environments larger than 100x100 meters, consider deploying additional robots or segmenting the environment into smaller manageable areas. For optimal performance, environments around 100x100 meters provide a good balance between search efficiency and area coverage.
5. *Enhanced Local Search:* Develop a more efficient local search strategy beyond the Random Walk Algorithm (RWA) to improve detection rates within local clusters. This can involve using heuristic or machine learning-based approaches to optimize the search paths dynamically.
6. *Redundancy Reduction:* Implement mechanisms to prevent redundant exploration of already searched areas. Techniques such as virtual stigmergy or communication protocols that share explored regions among robots can significantly reduce overall search time and increase efficiency.
7. *Obstacle Avoidance:* Ensure robust obstacle avoidance mechanisms are in place. The use of multiple proximity sensors and adaptive movement strategies based on sensor input can enhance the robots' ability to navigate complex environments effectively.

**5. Conclusion and future work.** Our focus in this study lies in the dispersion of robot groups achieved through an implicit division of the environment based on the positions of each group. To address this, we proposed a multi-robot search and rescue algorithm, PeSOA (Penguin-inspired Search Optimization Algorithm). The algorithm has been successfully implemented within the ARGoS simulation platform, and the obtained results are promising.

Future perspectives may include: (1) Further analysis is needed to evaluate the robustness of the proposed algorithm under different scenarios, such as varying cluster densities, uneven distribution of victims, or obstacles in the search area, (2) developing a more efficient local search strategy that surpasses the randomness of the current approach, (3) preventing redundant exploration of previously explored areas and consequently reducing the overall search time.

## REFERENCES


[1] Gheraibia, Y., Moussaoui, A. *Penguins search optimization algorithm (PeSOA).* In Recent Trends in Applied Artificial Intelligence: 26th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2013, Amsterdam, The Netherlands, June 17-21, 2013. pp. 222-231. Springer Berlin Heidelberg.

[2] Mansouri, A., Aminnejad, B., Ahmadi, H.. *Introducing modified version of penguins search optimization algorithm (PeSOA) and its application in optimal operation of reservoir systems.* Water Science and Technology: Water Supply, 18(4), pp. 1484-1496.

[3] A. Bo, J. Maoxin, X. Hanwen, X. Jiangling, W. Zhen, L. Benshuai and Z. Dan, *Coverage path planning for maritime search and rescue using reinforcement learning*, Ocean Engineering, 241 (2021), pp. 117–134.

[4] C. S. Won, P. H. Ji, L. Hanseob, S. D. Hyunchul and K. S. Young, *Coverage path planning for multiple unmanned aerial vehicles in maritime search and rescue operations*, Computers and Industrial Engineering, 161 (2021), pp. 107612.

[5] D. Ulrich, M. Bajestani, S. Ehsan, L. P. Yves and B. Giovanni, *Search and rescue with sparsely connected swarms*, Autonomous Robots, 47 (2023), pp. 849–863.

[6] D. Yuwen, *Multi UAV Search and Rescue with Enhanced A â^— Algorithm Path Planning in 3D Environment*, International Journal of Aerospace Engineering, 2023 (2023).

[7] G. Vikram, T. Ritu and S. Anupam, *Comparative Analysis of Fruit Fly-Inspired Multi-Robot Cooperative Algorithm for Target Search and Rescue*, in IEEE World Conference on Applied Intelligence and Computing, AIC 2022, pp. 444–450.

[8] N. Gomez, N. Pena, S. Rincon, S. Amaya, and J. Calderon, *Leader-follower Behavior in Multi-agent Systems for Search and Rescue Based on PSO Approach, in IEEE SOUTHEASTCON Conference, 2022, pp. 413–420.*

*[9]* H. Jie, S. Quanjun and X. Zhannan, Multi robot cooperative rescue based on two-stage task allocation algorithm*, Journal of Physics: Conference Series, 2310 (2022).*

*[10]* R. Kumar, M. Firthous and R. Kumar, Multiple oriented robots for search and rescue operations, in IOP Conference Series: Materials Science and Engineering, 912(2022).

[11] M. Michael, A. Z. Irène and Q. C Guy, *Ant colony optimization for path planning in search and rescue operations*, European Journal of Operational Research, 305 (2023), pp. 53–63.

[12] Paez, D., Romero, J. P., Noriega, B., Cardona, G. A., and Calderon, J. M., *Distributed particle swarm optimization for multi-robot system in search and rescue operations*, IFAC-PapersOnLine, 54 (2021), pp. 1–6.

[13] Rahman, A., Bhattacharya, A., Ramachandran, T., Mukherjee, S., Sharma, H., Fujimoto, T., and Chatterjee, *Adversearch and rescue: Adversearch and rescueial Search and Rescue via Multi-Agent Reinforcement Learning*, IEEE International Symposium on Technologies for Homeland Security, HST 2022.

[14] Ruetten, L., Regis, P. A., Feil-Seifer, D., and Sengupta, S., *Area-Optimized UAV Swarm Network for Search and Rescue Operations*, 10th Annual Computing and Communication Workshop and Conference, CCWC 2020, pp. 613–618.

[15] Wang, Y., Liu, W., Liu, J., and Sun, C., *Cooperative USVâ€“UAV marine search and rescue with visual navigation and reinforcement learning-based control*, ISA transactions, 137 (2023), pp. 222–235.

[16] Yang, L., Xing, B., Li, C., and Wang, W. , *Research on Artificial Bee Colony Method Based Complete Coverage Path Planning Algorithm for Search and Rescue Robot*, In 2022 5th International Symposium on Autonomous Systems (ISAS), pp. 1–6.


*Edited by:* Katarzyna Wasielewska-Michniewska
Regular paper
*Received:* Mar 18, 2024
*Accepted:* Jul 29, 2024