Modification and training of the “ant algorithm” for planning swarm operations of moving object

Y A Mostovoi¹, V A Berdnikov¹

¹Samara National Research University, Moskovskoe shosse 34A, Samara, Russia, 443086

Abstract. A swarm of mobile robots, as a system of relatively simple interconnected controlled objects, performs a common task simultaneously and in a distributed manner. When planning swarm operations associated with the creation in the service area through the frontal band of the zones of operation of the target equipment there is a need to consider the possibility of its operational regrouping, since at the time of planning the exact purpose of the swarm operation has not yet been determined, or is a secret, or is determined by a number of random circumstances. The execution of the swarm operation is advisable to carry out in two phases. The first phase starts even before the resolution of these uncertainties by creating a basic random network with a relatively small concentration of robots. In the second phase, after resolving the uncertainty, a programmable percolation route is formed by local rearrangement of the swarm objects, which provides a predetermined coverage of the target equipment of the robots with a certain service zone. A modified ant algorithm with evolutionary learning is considered in order to quickly solve the swarm problem of two-phase operation. In this case, you can significantly reduce the time for the operation.

1. Introduction
One of the fast-growing directions of robotics is group robotics of moving objects (or swarm robotics) [1], [5], [9]-[11], [20]. A robot swarm being the system of interconnected robots has a number of advantages most critical of which as compared to a conventional single robot are as follows:

- fast responses and range of action of such system due to the swarm distribution over the entire territory of the service zone;
- high probability of the assigned task fulfilment due to a possibility of redistributing goals and replacing the failed robot with another one from the swarm;
- variety of options to achieve the system goals by redistributing the roles between the swarm objects;
- simple solving of tasks by each robot which allows solving rather complex computing tasks when dealing with dozens of the same swarm objects.

The swarm operations related to the creation of a through frontal band composed of the mated zones of target equipment operation – swarm object payload (equipment for search, observation, communication, fire fighting, impact on objects in the service zone, etc.) are considered. In this case the geometrical interpretation of such operation is the creation of a through route or a route via the service zone [6], [15]-[18].
This task of swarm robotics fits well with percolation theory tasks. The percolation theory detects the through “conductive” route – percolation route formed in a random environment when the concentration of “conductive objects” increases and reaches the critical value.

The classical percolation theory considers the matrix with random filling as the model of random operating environment model in direct geometrical interpretation [3], [12], [14], [15], [21], [23]-[25]. In this square matrix with the number of lines \( L \), the random part of cells is “black”. It conducts a liquid or gas flow, transport or information flow. Other cells are “white”, which do not conduct the flow. As the concentration (probability of occurrence) of the black cells increases, some of their ribs start osculating in a random manner, which can be interpreted as the occurrence of close interaction, and merge. The osculating ribs of the “black” cells form random conductive clusters, which are generated and grow as the concentration of the “black” cells increases [14], [25].

The size of the square cell in the matrix in the task under consideration is determined by the circle inscribed into the cell, which is the range of the target equipment operation for the swarm object and results from the geometrical meaning of the “cluster” notion as the formation of close (pass-free) interaction between the objects located in the cells and osculating ribs in the author’s model of the service zone – matrix.

The classical percolation theory [14], [23]-[25] finds the concentration of the swarm objects in the service zone – “black” cells \( p_b \) – stochastic percolation threshold when the through random route is formed over the “black” cells via the entire matrix in the set direction – stochastic percolation cluster. However, the stochastic percolation cluster has a loose structure, multiple “dead branches” and is fairly excessive in terms of the quantity of required swarm objects.

The well studied classical percolation theory for flat grids for the task under consideration is impractical and the authors develop the programmable (or artificial) percolation theory. The through percolation route in this theory [6]-[8], [15]-[18] is formed not due to the increase in the concentration of “conductive black cells” down to the occurrence of the stochastic route concentration [9] known for the flat grid but is organized actively due to filling of intervals between the small clusters occurring on the matrix and supplementing the planned route with the active swarm objects.

When implementing the programmable percolation [6]-[8], [15]-[18], the preliminary stochastic base is created at the first phase using the objects that are randomly distributed in the service zone, with their concentration values much lower than the stochastic percolation threshold; the through percolation route is built at the second phase by introducing (installing) additional objects into the available inter-cluster intervals (see Figure 1). In this case the stochastic base concentration is selected so that total costs for the two-phase operation are minimum. [8] shows that the minimum costs for the two-phase operation occur when the concentration is about 0.2–0.25.

Figure 1. Example of solving the task of the two-phase operation task on the matrix with different concentrations: a) concentration \( p = 0.25 \), b) concentration \( p = 0.4 \).
2. Task setting

A robot swarm is considered. Each robot can interact with the neighbouring ones in the process of solving a certain task. The maximum quantity of clusters in the object swarm allows their easy commutation and combination for joint solving of tasks in any required place in the service zone. In the course of this model study in [6]-[8], [15]-[18] the following statistic phenomena have been found:

The first phenomenon found as a result of numerical statistic modelling [6], [7], [15]-[18] is the typical dependence of the average number of the clusters formed on the concentration, with the maximum value when the concentration is about 0.25. Such value of the concentration can be explained physically: as the concentration increases, the clusters grow and start combining actively, but their quantity decreases (see Figure 2).

![Figure 2. Average quantity of the clusters as the function of object presence probability in the cell – concentration resulted from the statistic modelling on matrices of different sizes; diagram 1 – normalized function of the average quantity of clusters, diagram 2 – function of the average quantity of clusters for the matrices 50 × 50, diagram 3 – function of the average quantity of clusters for the matrices 100 × 100, and diagram 4 – function of the average quantity of clusters for the matrices 150 × 150.](image)

The second statistic phenomenon is the presence of the maximum length of the programmable percolation route at the stochastic percolation route concentration. Here the maximum tortuosity of the route is observed and the quantity of the cells added to the programmable percolation route is zero.

In the course of the extensive mathematical experiment to study the two-phase operation [6], [8], [15]-[18], the functions of the average route length have been set for the programmable percolation from the concentration and the average quantity of the objects added to the programmable percolation route from the concentration (see Figure 3 and 4). This experiment was carried out as follows: first the objects with different concentrations were inoculated on a set of percolation matrices. Then the Hoshen-Kopelman algorithm [4], [13] was used to define the clusters; this was followed by building an optimal (with the minimum quantity of the cells added) programmable percolation route (the experiment was conducted for the matrices 50 × 50, 100 × 100, and 200 × 200 in size). After that the Lightning-Dijkstra’s algorithm [6]-[8], [15]-[18] was used to set multiple suspicious routes for the programmable percolation route and the best route was selected based on the criterion of the minimum quantity of the objects added.

When considering the solving of the programmable percolation task by means of a network of interconnected robots, which is called the robot swarm by us, the coherence of separate object functioning in the object system that causes the object system behaviour as the entire one is required. Taking into account a potentially large quantity of objects in the swarm (several hundreds and more), the processes of functioning control and the frontal structures of the swarm occurring in this case must
be the consequence of the swarm self-organization without “master arm” acting on each object of the swarm from outside. In this case the external goal setting is possible. It must be brought to each member of the swarm.

**Figure 3.** Diagram of the function $L(\rho)$ of the average programmable percolation route length.

**Figure 4.** Diagram of the function $R(\rho)$ of the average quantity of the objects added to the inter-cluster intervals of the programmable percolation route – cells added to the programmable percolation route.

To implement the self-organizing behaviour, each object of the swarm must have the control system that allows it to occupy the set position in the space. To do this, each swarm member must have its own position with help of the navigation subsystem. This position must be transferred to other swarm objects via the communication subsystem. So, each swarm member may have a binary matrix of the swarm formed in the on-board central computer as the model of the service zone where the value 1 means that the swarm member is set in this point of the space – matrix cell and 0 means that the swarm object is absent in this point of the space. After forming this matrix and setting the programmable percolation route, each swarm object “learns its manoeuvre” from the set point of the operating environment in the on-board central computers and occupies independently the required position.

However, restricted computing resources of each robot make it difficult to use the described approach to solving the two-phase operation tasks, since the use of the Lightning-Dijkstra’s algorithm requires considerable computing costs. As a result, the goal of this article is to create an algorithm which allows forming the shortest route in the set direction via the matrix with random filling, starting from any boundary cell of the matrix that passes through the available associated clusters, with the minimum quantity of the objects added to the inter-cluster intervals – “red cells” – and has less time of operation under other equal conditions as against the Dijkstra’s algorithm.

To solve this task, first, it is proposed to use the “ant colony algorithm” (see the explanation of the basic principles of this algorithm selection in [6]) and second, conduct a preliminary training of the robot control system using a set of matrices of random sizes and random concentration. This training
must improve the ant algorithm parameters and reduce the “ant colony”, which will be of great advantage for speed with the acceptable loss of accuracy.

3. Study of the operating speed of the programmable percolation route computing algorithm

The use of the Dijkstra’s algorithm [2] as the basic algorithm for searching minimum routes on the percolation matrix requires sufficiently large resources [6], [22]. Any known algorithm for this problem requires modification. So for Dijkstra’s algorithm, it is a modification of the cost of weights, which will depend on the size of the matrix. The average operating speed of the Lightning-Dijkstra’s algorithm is determined as \(O(L^5)\), where \(L\) is the percolation matrix size. This can be easily demonstrated as follows:

The average speed of the Dijkstra’s algorithm is estimated as \(O(n^2)\), where \(n\) is the number of nodes in the graph, away for the matrix (percolation binary matrix can be viewed as a graph with number of nodes \(L^2\)) is average speed – \(O(L^4)\). To select the average minimum route for the matrix, you need to run this algorithm from each point in the upper line to the lower line of the matrix, i.e. the average speed of the algorithm will be equal to \(O(L^5)\).

If fast sorting or binary bulk is used inside the Dijkstra’s algorithm, the speed of the algorithm under consideration here can be improved to \(O(L^4 \log L)\).

It is evident that the mass use of such algorithm will take sufficiently much time and require serious processor capacities; therefore, it was proposed to use the ant algorithm in [6].

In this case, any iterative algorithm will iterate over the full matrix \(L\) times, which will negatively effect the operation time of any such algorithm, regardless of its optimality, and this iteration is necessary for solving the problem. Therefore, it is necessary to use algorithms that do not require a large iteration of the graph nodes and, in the best case build in one pass, will immediately, a pseudo-optimal route, such as the first pass of the “ant” from the ant algorithm (in the further work of the ant algorithm, the search of paths is performed by members of the colony in order to clarify and improve the path; this iteration also needs to be reduced through to modifications of this algorithm).

It is evident that the use of the standard ant algorithm [22] on the percolation matrix is difficult because the algorithm selects the route in an equally probable manner at the first phase. To increase the speed of the algorithm convergence, it was necessary to correct the selection probability, i.e. add a specific modifier \(m_{ij}\) to the probability of the cluster selection and movement in the programmable percolation route direction. The remaining body of the classical ant algorithm was not changed [6], [22]. So, the formula for selecting the ant movement direction became as follows:

\[
p_k = \frac{p + m_{ij} + p_{hij}(n) + t\delta(i)}{\sum_{k=1}^{m} p + m_{ij} + p_{hij}(n) + t\delta(i)}
\]

where \(p\) is the route selection probability which is set a random manner \((p \in [0,1])\); \(m_{ij}\) is the cluster modifier which is equal to a certain random number if the matrix cell belongs to the cluster and 0 in other cases; \(p_{hij}(n)\) is the pheromone value \((p_{hij}(n) = p_{hij}(n - 1) - \Delta \ast L \ast L(P), \Delta\) is the pheromone evaporation rate; \(n\) is the iteration number); \(t\) is the modifier of the movement towards the percolation; \(\delta(i)\) is the delta function from \(i\), which is equal to \(\delta(i) = \begin{cases} 1, & i = i + 1 \\ 0, & \text{else} \end{cases}\); \(k\) is one of the possible directions of percolation (down, right, left); \(y(n) = \begin{cases} 1, & p_{hij} \geq 0 \\ 0, & \text{else} \end{cases}\), \(i\) is the number of the matrix line; \(j\) is the number of the matrix column.

The maximum speed (i.e. the case when the selected route passes through all the matrix cells) of this algorithm can be determined as \(O(L^3K)\), where \(L\) is the matrix size, and \(K\) is the quantity of ants. The operating speed of the ant algorithm can be assessed as follows, on an average (i.e. with well
selected parameters) $O(KL^2L(p))$. The comparison of the mean time of the algorithm operation implemented with the help of the C++ language is shown in Figure 5 and 6. All the results are for the processor Intel Core i7 CPU 870.

![Figure 5. Mean time of the algorithm operation.](image)

![Figure 6. Mean time of the ant algorithm operation for 100 “ants” produced from one cell on the matrix.](image)

The ant algorithm, however, depends heavily on the quality of parameter selection (pheromone evaporation rate, initial value of the pheromone and quantity of the pheromone added at each step, and colony size), which have a considerable impact on the “randomness” of the algorithm behaviour. So, the routes can not converge in case of the low value of the pheromone (the colony finishes) and the probability of incorrect route selection increases at the high value. The parameter selection itself is a difficult and unclear task due to a relatively large quantity of these parameters and their unclear interaction [22]. In this case it will be convenient to use the evolutionary approach to the parameter selection.
4. Evolutionary training of the ant algorithm

The ant algorithm body itself will be used as a phenotype. This means no changes to be made to this algorithm operation and the rule for selecting the optimal route [19].

The following parameters will be used as the genotype: \( m_{ij} \) (cluster modifier), \( p_{hij} \) (pheromone value), \( t \) (modifier of the movement along the percolation route), \( \Delta \) (pheromone evaporation rate), and \( K \) (quantity of ants) (see expression 1).

The training rule is the minimization of the following error:

\[
\xi = 0.1 \left( \frac{L_{\text{real}} - L(p)L}{L} \right) + 0.9 \left( \frac{R_{\text{real}} - R(p)L}{L} \right)
\]

where \( L_{\text{real}} \) is the obtained length of the programmable percolation path; \( L(p) \) is the reference normalized by \( L \) length of the programmed percolation path, obtained analytically; \( R_{\text{real}} \) is the obtained number of added cells; \( R(p) \) is the reference normalized by \( L \) number of added cells, obtained analytically; \( L \) is the matrix size.

The training of the genetic algorithm was conducted in accordance with the following rule:

1. Carrying out a two-phase operation based on the parent colony, computing its error \( \xi_m \);
2. Creating descendants based on the parent colony with introduced genetic mutations (random changes were made to the values of the genotype parameters), computing errors of descendants \( \xi_{e_i} \), where \( i \) is the descendant number;
3. Selecting the colony (parent or any of the descendants) which training error \( \xi \) is minimum.

The training convergence diagram is shown in Figure 7 and 8 for different training conditions: in the first case, the training of the modified ant algorithm was conducted for the fixed concentration

![Figure 7](image)

**Figure 7.** Dynamics in the error of the ant algorithm training using the evolutionary algorithm (fixed concentration, random size of the matrix).

![Figure 8](image)

**Figure 8.** Dynamics in the error of the ant algorithm training using the evolutionary algorithm (random concentration, random size of the matrix).
(\(p = 0.25\)) on the set of matrices of random size; in the second case, the training was conducted on the set of random matrices of random size, with the random concentration of the swarm objects. In both cases, the training was terminated starting from the 96\textsuperscript{th} generation (the same numbers of the generations during the training are nothing more than the coincidence), the error noise stabilized in accordance with the dispersions for the functions of the average programmable percolation length \((L(p))\) and average quantity of the objects added \((R(p))\), which are reference in expression 2.

It was also noted during the training that the ant algorithm, starting from a certain step (generation), almost stopped depending on the quantity of ants in the colony, i. e. it “evolved” to the standard quasi-optimal algorithm for searching the optimal route. This allowed reducing the quantity of ants to 3 in the colony and increasing considerably the speed of this trained algorithm operation. The selection of such quantity of agents in the colony is determined by the following reasons: probability of selecting the cell visited by the previous one (as this was noted in the course of the experiment) tends to one already at the third step for the trained heuristic parameters, i. e.:

\[
\begin{align*}
\sum_{k=1}^{3} p + m_{ij} + ph_{ij} + t\delta(i) & \rightarrow 1, \quad p \in [0,1], ph_{ij} > 0 \\
\sum_{k=1}^{3} p + m_{ij} + (ph - 2\Delta * L * L(P))r(i) + t\delta(i) & = 1
\end{align*}
\]

The mean speed of the trained ant algorithm is determined as \(O(3L^2L(p)) \approx O(L^2L(p))\). The training result is shown in Figure 9.

While analysing this diagram, it may be stated that the average error of the untrained ant algorithm is much higher than that in the Dijkstra’s algorithm and trained modified ant algorithm. In the course of the evolution process, the ant algorithm became almost ideally coinciding with the results of the Dijkstra’s algorithm and also became considerably advantageous in terms of speed, i. e. the quantity of

![Figure 9. Average quantity of the objects added by the considered algorithms.](image-url)
ants in the colony was reduced sharply. However, there is always a probability that the trained ant algorithm will initially select a non-optimal route. The long-term training may only reduce the probability of such outcome but not exclude it.

5. Conclusions
As a result of the completed work, the following conclusions can be made:

1. The possible large quantity of objects in the swarm (several hundreds and more), complex processes of the target swarm functioning control, with the formation of different frontal structures must be the consequence of the swarm self-organization without “master arm” acting on each object of the swarm from outside. This requires an increased time efficiency of operation of the programmable percolation route setting algorithm which cannot be ensured by the Lightning-Dijkstra’s algorithm.

2. The speed of the Lightning-Dijkstra’s algorithm was studied to solve the task of forming the through route of programmable percolation via the matrix with random filling. The adequate option of its replacement with the modified ant algorithm, with the maintained acceptable accuracy of output results was proposed.

3. The modified genetic ant algorithm was developed specifically for the task under consideration; the evolutionary method for its training was proposed and the training was conducted. This resulted in the considerable increase in the modified ant algorithm operation due to the decrease in the quantity of ants in the colony (evolution result) and set its parameters (pheromone evaporation rate, initial value of the pheromone, quantity of the pheromone added at each phase, and additional modifiers described in expression 1).

6. References
[1] Abraham A, Grosan C and Ramos V 2006 Swarm Intelligence in Data Mining Studies in Computational Intelligence.
[2] Agafonov A A, Myasnikov V V and Yumaganov A S 2018 Big data analysis in a geoinformatic problem of short-term traffic flow forecasting based on a k nearest neighbors method Computer Optics 42(6) 1101-1111 DOI: 10.18287/2412-6179-2018-42-6-1101-1111.
[3] Alexandrowicz Z 1980 Critically branched chains and percolation clusters Physics Letters A 80(4) 284-286.
[4] Babalievski F 1998 Cluster counting: the Hoshen-Kopelman algorithm vs. Spanning three approach International Journal of Modern Physics C 9(1) 43-61.
[5] Beni G 2004 From Swarm Intelligence to Swarm Robotics Swarm Robotics SAB Revised Selected Papers (Lecture Notes in Computer Science) 3342 1-10.
[6] Berdnikov V A and Mostovoi Y A 2017 Ant Algorithms in Planning Two-phase Operations in Large Networks Problems of Control and Modeling in Complex Systems 445-454.
[7] Berdnikov V A and Mostovoi Y A 2018 Analytical and numerical modeling of the process for cluster emergence of objects in a random environment Journal of Physics: Conference Series 1096.
[8] Berdnikov V A and Mostovoi Y A 2018 Statistical modeling of a scale network of nanosatellites Journal of Physics: Conference Series 1096.
[9] Bonabeau E and Meyer Ch 2001 Swarm Intelligence: A Whole New Way to Think About Business Harvard Business Review 79(5) 106-114.
[10] Gaiduk A R, Kaliaev I A and Kapustian S G 2009 Models and Algorithms of Collective Control in Robot Groups (Moscow: Phizmatlit). 
[11] Gaiduk A R and Kaliaev I A 2004 Swarm Principles of Control in the Group of Objects Mechatronics, Automation, and Control 12 27-38.
[12] Galam S and Mauger A 1996 Universal formulas for percolation thresholds Phys. Rev. E 53(3) 2177-2181.
[13] Hoshen J and Kopelman R 1976 Phys. Rev. B 14 3438-3445.
[14] Moskalev P V and Shitov V V 2007 Mathematical Modelling of Porous Structures (Moscow: Phizmatlit) 120.
[15] Mostovoi Y A 2011 Statistic Phenomena of Large Distributed Clusters of Nanosatellites Vestnik of S. P. Korolev Samara State University (National Research University) 26(2) 80-89.
[16] Mostovoi Y A 2013 Two-phase Operations in Large Networks of Nanosatellites Computer Optics 37(1) 120-130.
[17] Mostovoi Y A 2013 Controllable Percolation and Optimal Two-phase Operations in Large Networks of Nanosatellites Infocommunication Technologies 11(1) 53-62.
[18] Mostovoi Y A 2015 Modelling of Optimal Two-phase Operation in Random Operating Environments Automatic 51(3) 35-41.
[19] Pilinskii M, Rutkovskaia D and Rutkovskii L 2008 Neuron Networks, Genetic Algorithms and Unclear Systems (M.: Goriachaia linii – Telekom) 452.
[20] Sahin E 2005 Swarm robotics: From sources of inspiration to domains of application Swarm Robotics. Lecture Notes in Computer Science 3342 10-20.
[21] Sarshar N 2004 Scalable Percolation Search in Power Law Networks Proceedings of the Fourth International Conference on Peer-to-Peer Computing Zurich Switzerland.
[22] Shtovba S D 2003 Ant Algorithms Exponenta Pro. Mathematics in Applications 4 70-75.
[23] Stauffer D 1979 Scaling theory of percolation clusters Physics Reports 54 74.
[24] Stauffer D and Aharony A 1992 Introduction to Percolation Theory (London: Taylor & Francis) 179.
[25] Tarasevich Iu Iu 2002 Percolation: Theory, Applications, Algorithms (Moscow: USSR) 109.