ACTA IMEKO
ISSN: 2221-870X
December 2024, Volume 13, Number 4, 1 - 10

Path planning for multiagent system in a sensing field with obstacles and multiple base stations

Sára Szénási[1], István Harmati[1]

[1] *Dept. of Control Engineering and Information Technology, Budapest University of Technology and Economics, Budapest, Hungary*

ABSTRACT
In large area data collection, wireless long-distance data transmission through sensor network reduces network lifetime due to large energy consumption. Therefore, this paper presents a path planning problem for a multiagent system to periodically visit a set of sensor nodes in a sensing field with obstacles while minimizing the data collection time. During data collection, all sensor nodes are visited in each period exactly once by the robots. At the end of one period the collected data is uploaded to one of the base stations. This paper proposes a new approach for constructing clusters and a visiting sequence of nodes, that each robot must visit. To design the clusters and the visiting sequence of nodes a new algorithm based on Ant Colony Optimization is developed. The path planning algorithms from one and multiple base stations are created. Simulations are conducted in various sensing fields with different numbers of agents, measurements are defined, and the results are analyzed. The created algorithms can be utilized for example, to monitor weather phenomena in an agricultural area or reconstructing the path of an intruder in a guarded area.

Section: RESEARCH PAPER

Keywords: path planning; mobile robots; obstacle avoidance; multiagent system; ant colony optimalization; data collection

Citation: S. Szénási, I. Harmati, Path planning for multiagent system in a sensing field with obstacles and multiple base stations, Acta IMEKO, vol. 13 (2024) no. 4, pp. 1-10. DOI: 10.21014/actaimeko.v13i4.1572

Section Editor: Zafar Taqvi, USA

Received May 18, 2023; **In final form** October 10, 2024; **Published** December 2024

Corresponding author: Sára Szénási, e-mail: olasz-szabo.sara@edu.bme.hu

1. INTRODUCTION

Nowadays, the need for continuous data collection in a specified area is more and more common. The simplest way for such data collection is using wireless sensor networks (WSN) [1], [2]. In most applications this consists of two parts: the base station and a large number of sensor nodes. Sensor nodes are collecting data from the surrounding environment and providing this data to the central sink using either a single-hop or multi-hop approach [3]. In typical application forms the sensor nodes are powered by batteries, which in difficult environmental conditions cannot be changed, like in cases of, e.g., a huge land, radioactive, toxic, remote locations, or battlefield surveillance [4]. Also, the data loss rate increases with the distance and every data transmission rate is associated with an energy consumption rate, which is modelled as a non-decreasing staircase function of the distance [5]. The remote data transmission consumes a significant amount of energy, and this deteriorates possible network lifetime. Since short distance data transmission is more reliable than long distance, utilizing mobile robots improves the data collection rate. In addition, using mobile sinks to collect data is more secure than transmitting it via long-distance multi-hop communication [6]. This may be important in some safety-critical fields, such as military applications, as well. For these reasons, the data transmission mode could be implemented using data collection robots [7], [8]. There are many applications of this technology in literature from recent years [9], [10], [11]. For example, a time-constrained data collection problem involving a network of ground sensors situated on uneven terrain was solved by an Unmanned Aerial Vehicle (UAV) [12]. The problem of enabling autonomous mobile robots to operate in unstructured terrain and environments, and collect data in a time-efficient manner, was also discussed by [13].

Viable path planning for data collection unicycle robots in a sensing field with obstacles is a problem raised and solved in [14]. In this issue, there is only one base station in a sensing field. The robots start from the base station and then together visit all sensor nodes exactly once and at the end of the tour return to the base station and upload the collected data. The energy consumption of robots depends on the distance travelled along the planned path. Therefore, the paths created have a direct impact on the delivery delay and energy consumption of the system. In a sensing field, there are obstacles as well, and the

robots must avoid colliding with them. To create path planning algorithms, it is necessary to determine the criteria for an adequate path. By definition, a viable path is smooth, collision-free with sensor nodes/base station and obstacles, closed, and provides enough contact time with all the sensor nodes [14]. The data collection is accomplished by unicycle Dubins-cars [15], which can only move with constant velocity and bounded angular velocity. As a result, they can move only on straight lines and turn with a bounded turning radius depending on the robot's movement and maximum angular velocity. The reasons for using Dubins cars are that they have the most restricted mode of movement. If a problem can be solved using them, it can be solved using other types of mobile robots. Additionally, their movement is similar to UAVs flying at a constant altitude. Due to the kinematic properties of the robots, the path must be smooth. All nodes are bounded with a visiting circle with the minimum turning radius of the unicycle robot. Moreover, all obstacles' convex hulls are bounded with a safety margin, which is necessary because, in the case of the shortest path, the robot should move along the boundary of the convex hull. The path must be closed because of the periodical data collection. The robot downloads data only when it moves around the visiting circle, so it makes round trips around the node until it collects all the data from the sensor node. During path planning, it is assumed that the locations of all nodes and obstacles, as well as the shapes of the obstacles, are known. Between two objects – the safety convex hull of obstacles or the visiting circles of nodes – there are always four defined tangents. However, any tangents that intersect other safety circles of nodes or the safety convex hull of obstacles are removed to prevent collisions. When the robot arrives at a node in a visiting circle on a tangent, it starts downloading data. It makes round trips within the district of the visiting circle until it collects all the data from the node, and then it leaves the visiting circle of the node on a tangent. So, a path consists of an adequate configuration of tangents and arcs around obstacles at the safety distance or visiting circles around nodes.

Because of the periodic data loading, we need to determine the visiting sequence of nodes by solving a Travelling Salesman Problem (TSP). In this problem, the salesman wants to find the shortest closed path given a set of customer cities, starting and ending in his hometown. Since the TSP is an NP-hard problem, as the number of nodes to visit increases, the possible permutations will increase exponentially. Therefore, the time required to find the exact solution for longer visiting sequences will become too long to be practical for real-world applications. To solve this problem, the heuristic Ant Colony Optimization (ACO) is used. Ant Colony Optimization has been inspired by the behaviour of real ant colonies, as ants can find the shortest path between the food source and their anthill. The main idea of ACO is the indirect communication between the agents based on pheromone trails. Artificial ants prefer choosing nodes that are close by and connected by edges with a lot of pheromone trails. In addition, unlike in nature, the artificial ants may possess memory or shared memory. This enables them to recall which nodes they have already visited and they can be controlled by the queen process. There are many different methods and ideas to solve TSP with use of Ant Colony Optimization [16].

Since the Dubins car can only move with bounded velocity, using just one robot for data collection is highly time-consuming and inefficient, particularly in case of a vast sensing field. So, real-time data collection is unfeasible. Therefore, we use a multi-agent system consisting of more than one Dubins-car for data collection. To solve this issue, we create a k-ACO algorithm that constructs k paths of approximately equal lengths for a variable number of robots. Hence, the data collection time can be minimized. Algorithms are created for path planning for robots starting and ending at one or multiple base stations. Utilizing multiple base stations can lead to non-intersecting sub-paths, resulting in shorter data collection time and lower energy consumption. The k clusters of nodes and their visiting sequences are determined based on Ant Colony Optimization.

This paper is organized in the following way. Section 2 presents the data collection problem. In Section 3, we summarize the basic method for creating viable path [14]. In Section 4, we solve the problem for multi-agent system with robots starting and ending at the same base node, as described in reference [17]. In Section 5, we present an algorithm for path planning for multi-agent system with robots starting and ending at predefined different base stations. Then in Section 6, three different measurements are described to analyse results. Finally, in Section 7, we describe the simulation results in several different sensing field using different number of robots.

2. PROBLEM STATEMENT

During the path planning for data collection, Dubins cars are used. The goal is to collect all data from the sensor nodes periodically. For data collection, a multi-agent system consisting of k agents is used. In a sensing field, there are n sensor nodes and n_B base stations. All sensor nodes store g_i, $i \in [1, n]$ data. In the sensing field, there are also m obstacles, which the robots must not collide with. The locations of all nodes and obstacles are known before path planning. During the data collection all robot starts and ends its sub-path at the appropriate base node. They collectively download data from all sensor nodes and upload all the collected data to the correct base station. In the case of one base station, all robots start and end their sub-path at the same node. Minimizing the data collection time is a crucial issue because the paths created directly impact the delivery delay and energy consumption of the system. So, in one period, all sensor nodes are visited only once. Definition 1 defines k-viable sub-paths based on [14].

Definition 1 *k*- viable sub-paths

A path P is k viable sub-paths if the following eight conditions are satisfied: 1) smooth and collision-free of 2) sensor nodes/base stations 3) and obstacles, 4) closed, 5) offers enough contact time to read/write all the data from/to sensor nodes/base stations, 6) all P_l , $l \in [1, k]$ sub-path start and end at the Base_l , 7) each sensor node is visited by only one viable sub-path, 8) the data collection time is minimized.

In the case of one base station $\text{Base}=\text{Base}_1=\text{Base}_2=\ldots=\text{Base}_k$, so all sub-paths start and end at the same node called Base. The planned path must be smooth due to the kinematic properties of the Dubins car. To ensure a smooth condition, we define the safety convex hull of obstacles as per Definition 2 and the visiting circles of nodes as per Definition 3. During the data collection the robot moves around the safety convex hull of obstacles or the visiting circles of nodes and the tangents between them. Since the Dubins car cannot switch its moving direction immediately, the planned path must also satisfy the heading constraint outlined in Definition 4, as well. The Dubins car's minimal turning radius is $R_{\min} = v/u_{\text{M}}$, so the robot can only turn with a radius R that satisfies $R > R_{\min}$. The optimal motion of a Dubins car involves turning with $R_{\min}$ radius or moving on a straight line

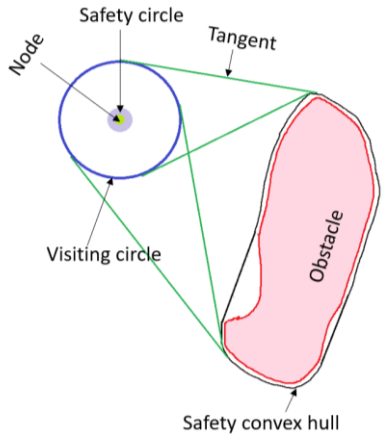


Figure 1. Illustrative examples for the nodes and their safety and visiting circles, the obstacle and their safety convex hull and the definition the common tangents between the objects.

[15]. Therefore, during data loading, the robot moves in a circle known as the visiting circle with R_{min} radius around the node.

Definition 2 Safety convex hull of obstacles

Any ∂o_j ($j \in [1, m]$) is a smooth curve with the curvature $c(p)$ at any point p satisfying $c(p) \leq \frac{1}{R_{\min}}$ and the minimum distance of the obstacle and the safety convex hull of obstacles is d_{safe}.

Definition 3 Visiting circle

The visiting circle is centred at the location of a sensor node/ base station with the radius of R_{min}.

Definition 4 Heading constraint

The heading constraint refers to that the robot's heading θ at the beginning of an edge in a Tangents Graph should be equal to that at the ending of the last edge.

The second condition of Definition 1 is that the planned path is collision free with nodes. It is defined as the safety circle of nodes that the planned path cannot intersect with. The third condition is satisfied when the path does not intersect any safety convex hull of obstacles.

Definition 5. Safety circle

The safety circle is centred at the location of a sensor node/ base station with the radius of d_{safe}.

In Figure 1 can be seen the illustration of the nodes denoted in light green their visiting and safety circles denoted in blue and light blue, the obstacles denoted in red and their convex hulls denoted in black. The four tangents between the objects are also presented with green in Figure 1.

Because of the periodic data collection, the planned path is closed. This requirement is met when lth sub-path starts and ends at the Base$_l$ base station. When there is only one base station all sub-path starts and ends at it.

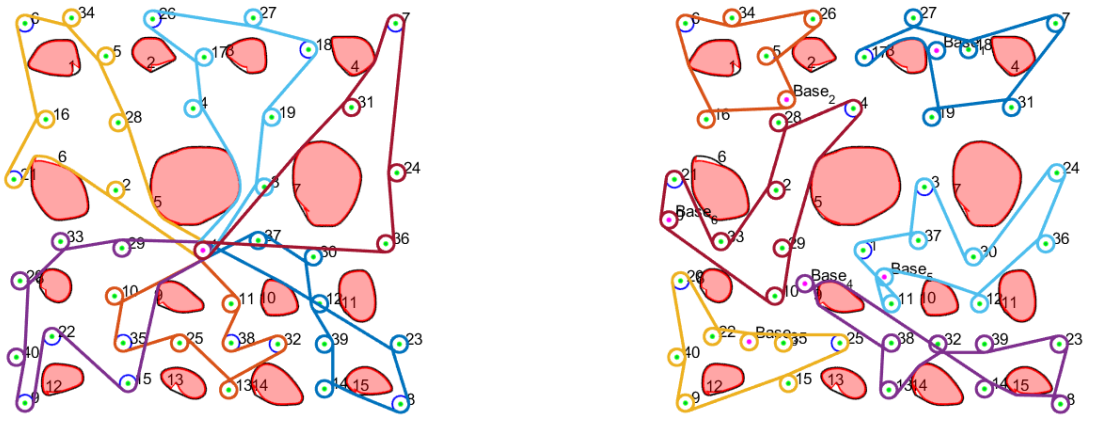
For the quality assurance of data transmission, the robot downloads data only when it moves along the visiting circle. To ensure sufficient contact time for data transmission, the robot rotates around the nodes on the visiting circle until the data transmission is completed.

The conditions 6), 7), and 8) can be provided by establishing the proper cluster and visiting sequence of nodes. This can be achieved, for example, using the ACO-based method.

Figure 2 shows illustrative examples for the k-viable sub-paths in scenarios with single and multiple base stations. The base stations are denoted by magenta, the sensor nodes are denoted by green, and their visiting circles are denoted by blue. The obstacles illustrated in with red and their safety convex hull are black. The different sub-paths of various robots are denoted by different colours. The length of individual sub-paths remains similar in the two setups, but the multiple base node setup has the advantage of fewer intersections of tangents used by the robots. Additionally, the malfunction of one base node has a much less serious impact. In the case of multiple base nodes, data can still be obtained from the other regions of the sensing field. However, this setup has the disadvantage that accessing data from the entire sensing field, requires an additional step of data collection from the individual base nodes.

3. SUMMARIZE SHORTEST VIABLE PATH PLANING ALGORITHM

In this section, a Shortest Viable Path Planning (SVPP) algorithm is introduced [14]. This creates a path for a single agent data collection that is smooth, collision-free with sensor nodes and obstacles, closed, and allows the robot to read all the data from the sensor nodes and upload it to the base station. The major steps of SVPP are summarized in Algorithm 1.



a) One base station

b) Multiple base stations

Figure 2. Illustrative examples for the k-viable sub-paths in case one and multiple base station.

Algorithm 1. Shortest Viable Path Planning (SVPP).

1. Compute permutations of nodes (Σ) without obstacles (tangents that intersect obstacles are available at this point) – TSP (Travelling Salesman Problem) with nodes.
2. Compute permutation (Σ') of nodes and obstacles by adding blocking obstacles to the permutation.
3. Simplify Tangent Graph $G(V, E)$ by keeping the edges and the vertices related to permutation Σ' and deleting others of $G'(V', E')$ Simplified Tangent Graph.
4. Convert the Simplified Tangent Graph $G'(V', E')$ to tree-like graph T.
5. Search the shortest path P in tree-like graph T depend on the initial configuration.
6. Compute Σ by solving ATSP instance based on $G(V, E)$
7. Compute Σ' by adding those obstacles to Σ that safety boundaries block tangents between nodes.
8. Simplify $G(V, E)$ to $G'(V', E')$ by keeping the edges and the vertices related to Σ' and deleting others.
9. Convert $G'(V', E')$ to tree-like graph T.
10. Given an initial configuration, search the shortest path P in T.

In the first step, the permutations of nodes without obstacles are determined by solving the Traveling Salesman Problem. Since there are always four common tangents between two visiting circles of nodes, the average length of tangents is used here. At this point, there can be tangents intersecting obstacles.

The second step of this algorithm involves adding the blocking obstacles to the permutation and constructing a Simplified Tangents Graph [18]. The Simplified Tangents Graph contains the common tangents that can be drawn between the sequentially visiting circles in the determined permutation of nodes. The tangent points have vertices, while the tangents and arcs that running between the tangent points serve as edges. When any obstacle blocks the route between any pair of visiting circles, the tangents passing the obstacles safety boundaries are also included in the Simplified Tangents Graph, and the algorithm inserts the obstacles into the permutation between the two nodes. The blocked tangents have been removed from the Simplified Tangents Graph. One obstacle can block more than one tangent.

In the next step, it converts the Simplified Tangents Graph to a tree-like graph. First, it assigns the tangent's direction because the unicycle robot cannot turn with a zero radius due to the robot' s movement being determined by a heading constraint. This means that the heading angle of the robot at the beginning of an edge - tangent or arc around the object - should be equal to that at the end of the last edge. It then creates graph nodes for each tangent point from the Simplified Tangents Graph. When the robot starts moving on a tangent, it can only arrive at one tangent point of the next object. Therefore, the tree-like graph consists of subgraphs corresponding to objects. If the tangents or the arc between two tangent points adhere to the heading constraint, they will be added to the tree-like graph.

Finally, the algorithm searches for the shortest path in both clockwise and counterclockwise directions, starting and ending from the base node while visiting the circle.

A more effective Generalized-SVPP algorithm [19] was developed based on the SVPP algorithm. The main steps are the same, but when running our G-SVPP algorithm, the tree-like graph contains more usable tangents due to the new methods for handling obstacles. Additionally, a new condition of blocked tangents by the nodes' safety or visiting circle is also presented. One can also find the algorithm for creating a tree-like graph from the Simplified Tangents Graph in [19]. In this paper, our G-SVPP algorithm [19] was used to plan viable sub-paths for the Dubins cars.

Algorithm 2. k-ACO.

1. Run Algorithm 3 for all nodes → sub-permutations $\Sigma_1, \ldots, \Sigma_k$
2. Run step 2-5 from G-SVPP algorithm [19] to permutations $\Sigma_1, \ldots, \Sigma_k \rightarrow$ sub-paths $P_1, \ldots, P_k$

4. DESIGN K-SUBPATHES USING ANT COLONY OPTIMALIZATION FROM ONE BASE STATION

First, the case of a sensing field with only a single base node was investigated. A new method was proposed to create *k*-viable sub-paths in the case of a single base station [17]. The steps of path creation are summarized in Algorithm 2. The first step involves determining the clusters and the visiting sequence of nodes by solving a Multiple Traveling Salesman Problem (MTSP) using Ant Colony Optimization. The second step involves creating viable paths from all sub-paths using the G-SVPP [19] algorithm.

First step is solving MTSP based on Ant Colony Optimization. There are many different methods to solve this problem [20], [21], [22], [23], [24]. In this paper, the basic idea is to create a single path that includes the base node repeated *k* times, where *k* denotes the number of agents [20]. A new method was also proposed to determine the clusters and sub-permutations of nodes. The main step of the solution is summarized in Algorithm 3.

In the first step, initializing d_{ij}, η_{ij}, τ_{ij} and determining the candidate list, as well as the minimum and maximum number of nodes between two base nodes, including the minimum and maximum elements of the sub-paths.

$$N_{\min} = \max\left\{\left\lfloor \frac{N}{0.75\,k} \right\rfloor, 3\right\}, \tag{1}$$

$$N_{\max} = \max\left\{\left\lfloor \frac{N + k - 1 - N_{\min}}{k - 1} \right\rfloor, 3\right\}, \tag{2}$$

where N denotes the number of cities. The sub-paths must contain a minimum of three nodes since closed paths are required. In this step, we use Euclidean distance to determine the distance d_{ij} between the *i*th and *j*th nodes ($i, j \in [1, N]$).

At the beginning, each of the *m* ants is located on a base node. When the ants choose the next node, they must pay attention to the minimum (1) and maximum (2) node numbers. After a sub-path reaches the minimum number of nodes, the base node can be selected again. Also, there is a maximum number of possible nodes in each sub-path. Upon reaching this maximum, the base node must be selected next. To visit all sensor nodes once and only once, a tabu list was constructed. This contains all sensor nodes that have already been visited for each sub-path. It is introduced a $\mathcal{N}_i^h$ set, which is the feasible neighbourhood of ant *h* located at node *i*, that is, the set of nodes which ant *h* has not yet visited. For better convergence, a candidate list can be used

Algorithm 3. ACO-MTSP.

1. Initialize $d_{ij}, \eta_{ij}, \tau_{ij}$, determine candidate list and calculate the minimal and maximum number of sub-path cities equations (1)-(2).
2. Update the pheromone trails about equations (5)-(6).
3. Place the ants at the base node.
4. for every h ($h \in [1, m]$) ant repeat:

   If the last base node has been chosen N_{max} node it must be the base node chosen. Then jump to the next iteration.

   else:

   If the last base node has been chosen N_{min} nodes, it can be the base node chosen.

   a. Calculate p_{ij}^h from the actual city i to every city j from equation (4)
   b. Select next node about q_0 random number or p_{ij}^h probabilities and tabu and candidate list (3).
   c. Update tabu list.
5. Repeat step 4 while all cities chosen.
6. All tour of ants cut off subtours.
7. For every subtour run 2-OPT Algorithm [26]
8. Determine the cost of the maximal sub-pathlength for all ants tour (8) and the sum of the subtour length for all ants (10).
9. Save the best tour.
10. Calculate $\Delta\tau_{ij}^h$ from equation (7)
11. Repeat step 2-10. until reaching maximum cycle number.
12. Choose the minimal cost tour from the best tours.

containing the $k_{candidate}$ closest nodes as candidates for each node. The effect of the candidate list occurs within the set $\mathcal{N}_i^h$

$$\mathcal{N}_i^h = \begin{cases} \mathcal{N}_i^{h(t)} \cap \mathcal{N}_i^{h(c)}, & \text{if } \mathcal{N}_i^{h(t)} \cap \mathcal{N}_i^{h(c)} \neq \emptyset \\ \mathcal{N}_i^{h(t)}, & \text{else} \end{cases}, \tag{3}$$

where $\mathcal{N}_i^{h(t)}$ set contains the nodes that are not in the tabu list and the $\mathcal{N}_i^{h(c)}$ set contains the nodes from the candidate list of node i.

It is defined as pheromone strength $\tau_{ij}(t)$ to each e_{ij} arc. This is a numerical information that is modified during algorithm's execution, where t represents the iteration counter. At each step, an ant h choice the next place based on probability function that depends on the pheromone trails and the distance from the other nodes. The probability that an ant h, currently positioned at node i, chooses to move to node j at the t-th iteration of the algorithm is

$$p_{ij}^h = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^h} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} \quad \text{if } j \in \mathcal{N}_i^h, \tag{4}$$

where $\eta_{ij} = 1/d_{ij}$ is a priori available value that depends on d_{ij} which represents the distance between node i and j. α and β are two parameters that determine the relative influence of the pheromone trail and the heuristic a priori information.

To converge to a global optimum, a randomly generated factor was created [25]. Determining $0 < q_0 < 1$ constant number and before the ant chooses the next node, the algorithm generates a random number $q \in [0, 1]$. If $q_0 < q$ the next node is selected randomly from the $\mathcal{N}_i^h$ set, otherwise, the next node is selected based on the calculated probability depending on the pheromone trails and the node distances (4).

After all ants have constructed their tours, the 2-OPT algorithm [26] can be used to optimize the neighbouring nodes within the tours. So, iterating through all nodes of tours and then exchanging the neighbouring nodes with each other. When the new tour is shorter than the previous one, the new tour is kept; if not, the previous version of the tour is used. After determining the length of the tours, the next step involves updating the pheromone trails. This is done by

$$\tau_{ij}^h(t+1) = (1-\rho)\tau_{ij}^h(t) + \Delta\tau_{ij}^h \tag{5}$$

$$\tau_{ij}(t+1) = \sum_{h=1}^{m} \tau_{ij}^h(t+1), \tag{6}$$

where $0 < \rho < 1$ is the possible value of the pheromone trail evaporation. The parameter ρ can be used to prevent uncontrolled growth of the pheromone trails left by the ants, allowing the algorithm to forget previous suboptimal routes. So, using the parameter ρ helps converge to the optimal solution instead of the local optimum. The $\Delta\tau_{ij}^h$ rate of the pheromone update on edge e_{ij} based on the h-th ant is determined

$$\Delta\tau_{ij}^h = \begin{cases} \frac{Q}{L_h} + \frac{Q_{best}}{L_{best}}, & \text{if } h = best \\ \frac{Q}{L_h}, & \text{else} \end{cases} \tag{7}$$

where $best$ denotes the shortest tour of the cycle, L_h the length of the h-th ant's tour, and L_{best} indicates the length of the best, shortest tour.

The cost function that must be minimized is the maximum length of sub-path lengths

$$L_{cost}^h = \max_i l_i^h \tag{8}$$

where l_i^h denotes the hth ant ith sub-pathlength.

The pheromone is updated by the full path length (sum of the sub-pathlength) and the cost

$$\Delta\tau_{ij}^h = \begin{cases} \frac{Q}{L_{sum}^h} + \frac{Q_{best}}{L_{cost}^h}, & \text{if } h = best \\ \frac{Q}{L_{sum}^h}, & \text{else} \end{cases} \tag{9}$$

where L_{sum}^h denotes the sum of the lengths of the hth ant

$$L_{sum}^h = \sum_{i=1}^{k} l_i^h. \tag{10}$$

The minimal cost tour for every ant is saved, and the algorithm is run for maximal cycle number times. After we choose the minimal cost solution from the best tours of iterations.

Algorithm 4. k-ACO from multiple base station.

1. Run Algorithm 5 for all nodes → sub-permutations Σ_1, …, Σ_k
2. Run step 2-5 from G-SVPP algorithm [19] to permutations Σ_1, …, Σ_k → sub-paths P_1, …, P_k

5. DESIGN K-SUBPATHES FROM MULTIPLE BASE STATIONS

In this section we investigate the case of multiple base stations in a sensing field. So, the different robots are start and end their tours from various base stations. Each robot has its own base station denoted by Base$_l$, $l \in [1, k]$. To design k-viable sub-paths from multiple base stations, a new method is proposed. The main step of path planning for robots starting from different base stations is summarized in Algorithm 4. The first step is determining the clusters and sub-permutations of sensor nodes is solved by ant colony optimization. Secondly, the G-SVPP [19] algorithm is executed to identify viable sub-paths.

The main steps for determining the clusters and permutations of sensor nodes are outlined in Algorithm 5. The main steps of Algorithm 5 are the same as than the main steps of Algorithm 3, but instead of using m ants and cutting off the tour into k subtours, it uses m ant teams, with each ant team consisting of k ants. Each ant builds a subtour immediately, and the ants of the same ant team visit all sensor nodes together. The ants of the same ant team build their sub-tours simultaneously, but they share the same tabu list. This guarantees that each sensor node will be visited exactly once.

Algorithm 5. ACO-MTSP from multiple base stations.

1. Initialize d_{ij}, η_{ij}, τ_{ij}, determine candidate list
2. Update the pheromone trails about equations (5)-(6).
3. Place each l, $l \in [1, k]$ ants at Base$_l$ node.
4. for every l, $l \in [1, k]$ ant of h ($h \in [1, m]$) ant team repeat:
   - If the tour of l ant is finished, then jump to the next iteration.
   - If the tour of l ant contains any sensor nodes then $\mathcal{N}_i^{h(t)} = \mathcal{N}_i^{h(t)} \cup$ Base$_l$
   - If all s, $s \in [1, k] \backslash l$ are finished their tour then $\mathcal{N}_i^{h(t)} = \mathcal{N}_i^{h(t)} \setminus$ Base$_l$
   - 4.1. Calculate p_{ij}^h from the actual city i to every city j from equation (4)
   - 4.2. Select next node about q_0 random number or p_{ij}^h probabilities and $\mathcal{N}_i^h$ (3).
   - 4.3. Update tabu list.
5. Repeat step 4 while all cities chosen.
6. For every subtour run 2-OPT Algorithm [26]
7. Determine the cost of the maximal sub-pathlength for all ants tour (8) and the sum of the subtour length for all ants (10).
8. Save the best tour.
9. Calculate $\Delta\tau_{ij}^h$ from equation (9)
10. Repeat step 2-9. until reaching maximum cycle number.
11. Choose the minimal cost tour from the best tours.

The first step of tour creation is placing all ants at the base stations, so all $l \in [1, k]$ ant of $h \in [1, m]$ ant team starts its tour at Base$_l$ node. The next node is chosen based on q random number same as Algorithm 3. Each subtour must contain at least one sensor node before it can choose the base station again. When all sensor nodes must be visited and all ants, except one finished their subtours, the remaining ant can select its base station again after all sensor nodes have been visited. After all subtours are finished, the 2-OPT algorithm [26] is applied to each subtour. The cost and length of tours are then determined. Finally, the algorithm saves the best tour of the cycle. After the iterations, the minimum cost tour from the saved tours is chosen.

6. MEASUREMENTS

Three different measurements [17] are used to evaluate the simulation results and to compare the different solutions based on the number of robots. First, the maximum access time of the sub-paths is analysed to determine the data collection time in the entire sensing field.

Second, the sum of the sub-path lengths was analyzed, which is proportional with the energy consumption of the robots during traversal.

Third, determining the normalized average of the absolute deviations of the length of sub-paths.

$$AVEDEV^{norm} = \frac{1}{k}\sum_{l=1}^{k}\frac{|l_l - \bar{l}|}{\bar{l}}, \tag{11}$$

where l_l denotes the sub-path length of l th robot and $\bar{l}$ denotes the average of the sub-path lengths. This shows the data delay difference between the nodes from the different agents sub-paths.

7. SIMULATION RESULTS

The simulations take place in a series of ten different 200 m × 200 m virtual sensing fields with a collection of 15 disjoint obstacles and 40 sensor nodes. The $k \in [2, 8]$ robots multiagent systems is used during the simulations. The Dubins car velocity is $v = 4$ m/s and the maximum angular velocity is $u_M = 1$ rad/s, therefore the minimum turning radius and the visiting circle's radius is $R_{min} = v/u_M = 4$ m. Each sensor node stores $g = 0.5$ MB data and the base node $g_B = (n_l - 1)$ 0.5 MB collected data is uploaded by the robot to their own base station, where n_l denotes the number of nodes of the l th ($l \in [1, k]$) sub-path. The data transmission rate at the visiting circle is $r = 250$ kB/s. The obstacle safety margin is $d_{safe} = 0.5$ m.

We use $k_{candidate} = 0.2\ N$ closest city to the candidate list. To determine the parameters, the classical parameters of [19]: $\alpha = 1$, $\beta = 2$, $\rho = 0.1$, $q_0 = 0.9$ and $Q = Q_{best} = 1$ was used. The cycle number is 2000, and the number of applied ants $m = N/2$.

First, the case of a single base station is presented, along with the results of Algorithm 2. In Figure 3 a) can be seen the data collection time using $k \in [2, 8]$ robots in ten different sensing fields is illustrated. The data collection time exponentially decreases with the number of agents. In the different sensing fields, the data collection time is approximately equal. The construction of the sensing field significantly influences the data collection time. For example, in Sensing Field 6, denoted as green, the data collection time is longer due to the location of the large obstacles compared to other sensing fields, and the rate of decrease is also slower. The sum of the sub-path lengths can be

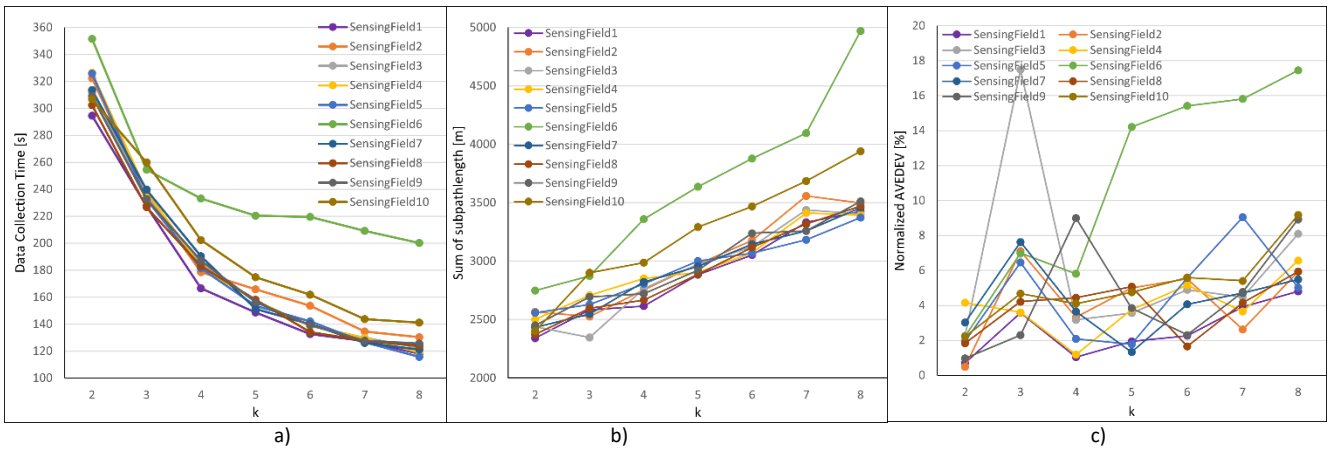


Figure 3. The simulation results of Algorithm 2 in ten different sensing field applied $k \in [2, 8]$ robots. The data collection time, sum of sub-pathlength and the normalized AVEDEV.

seen in Figure 3 b). This measurement approximately increases linearly with the number of applied agents. The issue arises from having only one base node, as each robot needs to return to this node. If the cluster nodes are distant from the base station, the robots will have to take longer trips. In Figure 3 c) can be seen the average of the absolute deviations of the sub-path lengths can be observed (11). Most of the cases involve less than 10 % of obstacles, but their location can significantly alter this. For example, in Sensing Field 6 applied $k \in [5, 8]$ the $AVEDEV^{norm}$ around equation (11) is 14-18 %. Summarily, as the number of agents increases, the data collection time exponentially decreases, and the energy consumption of the robots, which depends on the sum of the sub-path length, increases linearly. So, the optimal number of robots depends on the goal and the energy sources.

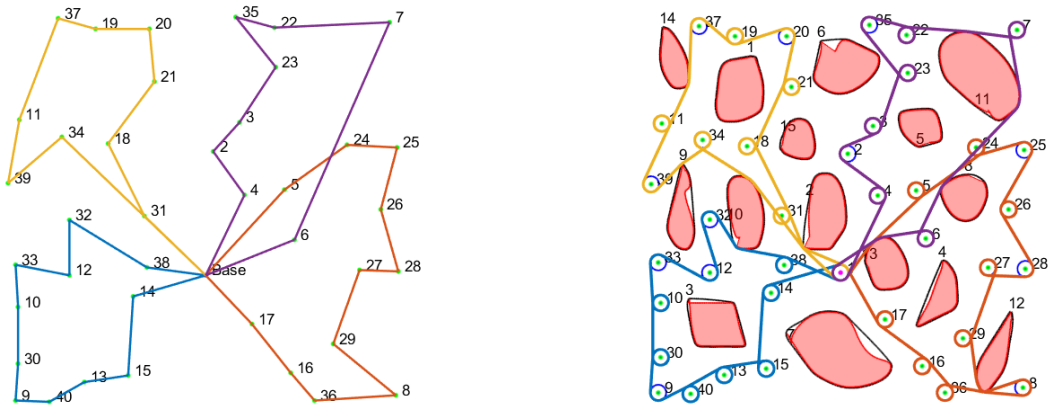
In Figure 4 can be seen the results of Algorithm 2 in Sensing Field 8 applied $k = 4$ robots. Figure 4 a) shows the path created by Algorithm 3, and Figure 4 b) displays the planned viable path of Algorithm 2. Table 1 shows the sub-path lengths of Figure 4. The path lengths in Figure 4 a) also include the length of data transmission.

In Figure 4 it can be seen that the first sub-path, denoted in blue is influenced by the obstacles at just one tangent, while the second sub-path, denoted in red is significantly longer due to Obstacle-12. The third sub-path, denoted in yellow is blocked by Obstacle-10, Obstacle-9, and Obstacle-14. The path circumvents these obstacles to find the shortest route. Consequently, the planned path can minimize the length needed to navigate around the visiting circles slightly. In the fourth sub-path, denoted by purple, between the Node-6 and Node-7, there are Obstacle-8 and Obstacle-11, which significantly increased the path length. On Table 1, it is shown that the planned sub-paths for the Dubins cars are approximately 19-27 % longer than the planned path of Algorithm 3. This difference is caused by the blocked obstacles and the additional path around the visiting circles.

Secondly, the simulated case of multiple base stations can be observed, to visualize the results of Algorithm 4. In this case, there are k base stations for the k agents. Each agent has its own base station, and these base stations are randomly placed to avoid obstacles and ensure free movement within the visiting circle configurations. In Figure 5 a) can be seen the data collection time

Table 1. The sub-path lengths of Figure 3.

The sub-path lengths of Figure 3 a)		The sub-path lengths of Figure 3 b)
505.2 m		632.5m
577.2 m		731.9 m
552.6 m		659.1 m
521.0 m		644.5 m



a) Step 1: Result of Algorithm 3

b) Step 2: Run GSVPP algorithm [19] for each sub-paths

Figure 4. The results of Algorithm 2 steps in Sensing Field 8 applied $k = 4$ agent.

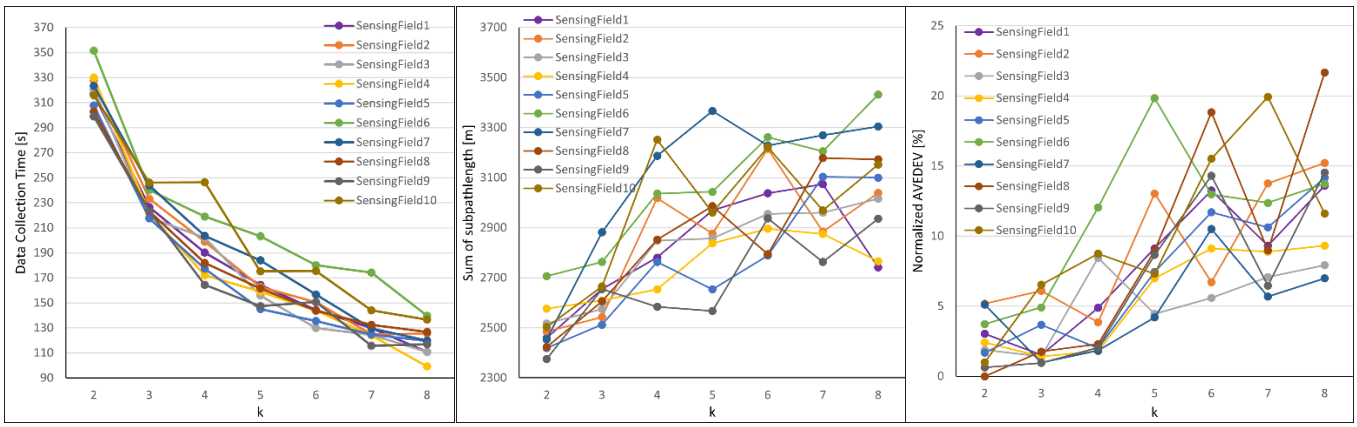


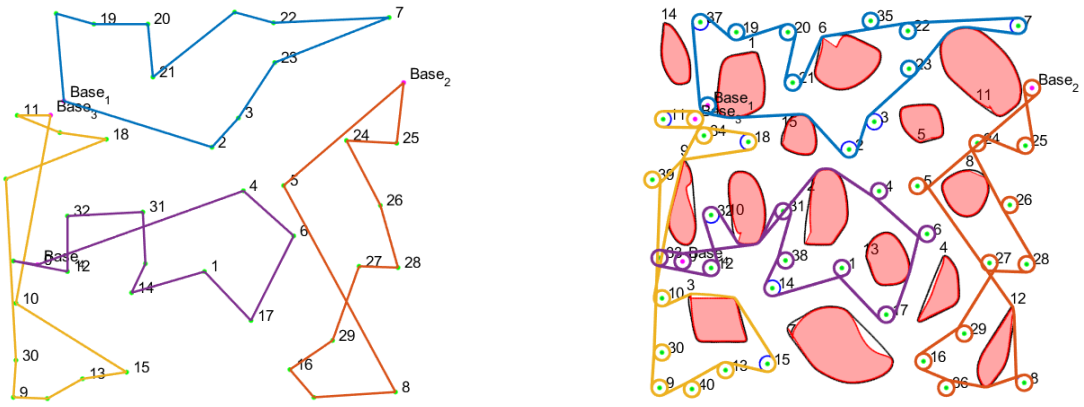
Figure 5. The simulation results of Algorithm 4 in ten different sensing field applied $k \in [2, 8]$ robots The data collection time, sum of sub-pathlength and the normalized AVEDEV.

using $k \in [2, 8]$ robots in ten different sensing fields, can be observed. The data collection time decreases exponentially with the number of agents. In the various sensing fields, the data collection time is approximately equal; however, the difference is greater than in the previous one base station case. The data collection time depends on the layout of the sensing field and the localizations of the base stations. Figure 5 a) shows the data collection times for the simulations with a single base station are shown. In this case, the data collection time for Sensing Field 6 is exceptionally larger than that of the other sensing fields due to the presence of a large obstacle. In the case of multiple base stations, the data collection time on Sensing Field 6 is just slightly longer than the data collection time on the other sensing fields. According to the locations of the base stations, sensing nodes, and obstacles, the data collection time for $k = 7$ and $k = 8$ agents can significantly decrease compared to the case with only one base station. For example, in Sensing Field 4, denoted as yellow, the sum of sub-path lengths can be seen in Figure 5 b). This measurement shows an increasing trend with the number of agents, but not linearly like in the case of a single base station. The sum of sub-path lengths is smaller than in the one base station case because the base stations are closer to the sensor nodes within the same cluster. This means that the required energy consumption is lower during the data collection. The normalized average of the absolute deviations of the length of sub-paths can be seen in Figure 5 c). This tendency increases with the number of robots used. Due to the random placement of the base stations, this measurement is larger than in the case of a single base station. It is possible that two or more base stations are placed close to each other while another one is far from them. In this case, the robots from the first group that start at the base station must visit sensor nodes that are far from them, and the robot that starts far from the other robot must visit more sensor nodes during its tour. Therefore, the lengths of the sub-paths and, consequently, the data collection delays of the various sensor node clusters will vary.

In Figure 6, the results of Algorithm 5 in Sensing Field 8 applied to $k = 4$ robots can be seen. Figure 6 a) shows the path created by Algorithm 5, while Figure 6 b) displays the planned viable path of Algorithm 4. Table 2 shows the sub-path lengths

Table 2. The sub-path lengths of Figure 5.

The sub-path lengths of Figure 5 a)		The sub-path lengths of Figure 5 b)
586.0 m		723.1m
610.7 m		728.0 m
593.2 m		721.5 m
557.1 m		679.8 m



a) Step 1: Result of Algorithm 5

b) Step 2: Run GSVPP algorithm [19] for each sub-paths

Figure 6. The results of Algorithm 4 steps in Sensing Field 8 applied $k = 4$ agent.

of Figure 6. The path lengths in Figure 6 a) also include the length of data transmission. The sub-paths are self-intersecting, which can result in more iterations being required when running Algorithm 5 based on ant colony optimization. The fourth sub-path, denoted by purple, is the most self-intersecting but also the shortest sub-path, so its length does not influence the data collection time. In addition, the first sub-path, denoted by blue, is not self-intersecting and is only 5 meters shorter than the second, longest sub-path, denoted by red. After removing loops from the second sub-path, the data collection time will not be significantly reduced. The obstacles blocked all sub-paths, so the length of the sub-paths became longer. Table 2 shows that the planned sub-paths for the Dubins car are approximately 19-23 % longer than the planned path of Algorithm 5. This difference is also caused by the blocked obstacles and the additional path around the visiting circles.

Summarizing, it is worth to choose as many robots as possible when favouring the minimalization of data collection time. It is possible to create a method to find a better location for base nodes.

CONCLUSIONS AND FUTURE PROSPECTS

In this paper, we investigated the challenges associated with planning the shortest path for a multiagent system utilizing Dubins cars with data loading capabilities. In both single and multiple base station cases were present. We propose new algorithms for identifying clusters of nodes and node permutations for path planning. We used heuristic Ant Colony Optimization. We run simulations for a varying number of robots in a multiagent system across different sensing fields. We compare the various solutions and scenarios. We utilize robots whenever possible to exclusively reduce data collection time. The location of the base stations significantly influences the data collection time and energy. It is suggested to develop a method to address this issue. Applying multiple base stations can result in a smaller data collection time. In future studies, we plan to introduce new path planning methods. For example, robots will be placed at specific points in the sensing field, and after collecting data in one direction, all robots will return to the base node. This application can be used, for example, in search and rescue operations after natural disasters to locate survivors. This application can be further enhanced by introducing priority nodes that the robots aim to reach first while still minimizing the overall path length.

ACKNOWLEDGEMENT

Supported by the **ÚNKP-22-2-III-BME-233** New National Excellence Program of the Ministry for Culture and Innovation from the source of the National Research, Development and Innovation Fund.

Project no. TKP2021-NVA-02 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

REFERENCES

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, J. Anderson, Wireless sensor networks for habitat monitoring, 1st ACM Int. Workshop on Wireless Sensor Networks and Applications, Atlanta, Georgia, USA, 28 September 2002, pp. 88-97. DOI: 10.1145/570738.570751

[2] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, B. Krogh, Energy-efficient surveillance system using wireless sensor networks, 2nd Int. Conf. on Mobile Systems, Applications, and Services, ACM, Boston, Massachusetts, USA, 6 – 9 June 2004, pp. 270-283. DOI: 10.1145/990064.990096

[3] J. N. Al-Karaki, A. E. Kamal, Routing techniques in wireless sensor networks: a survey, in IEEE Wireless Communications, vol. 11, no. 6, 2004, pp. 6-28. DOI: 10.1109/MWC.2004.1368893

[4] Y. Gu, F. Ren, Y. Ji, J. Li, The evolution of sink mobility management in wireless sensor networks: a survey, IEEE Commun. Surv. Tut. 18 (1), 2015, pp. 507–524. DOI: 10.1109/COMST.2015.2388779

[5] X. Ren, W. Liang, W. Xu, Data collection maximization in renewable sensor networks via time-slot scheduling, IEEE Trans. Comput. 64 (7), 2015, pp. 1870–1883. DOI: 10.1109/TC.2014.2349521

[6] Y. Gu, F. Ren, Y. Ji, J. Li, The evolution of sink mobility management in wireless sensor networks: a survey, IEEE Commun. Surv. Tut. Vol. 17, 2015. DOI: 10.1109/COMST.2015.2388779

[7] I. Chatzigiannakis, A. Kinalis, S. Nikoletseas, Sink mobility protocols for data collection in wireless sensor networks, 4th ACM Int. Workshop on Mobility Management and Wireless Access, ACM, Terromolinos, Spain, 2 October 2006, pp. 52-59. DOI: 10.1145/1164783.1164793

[8] Y. Yun, Y. Xia, Maximizing the lifetime of wireless sensor networks with mobile sink in delay-tolerant applications, IEEE Trans. Mob. Comput. Vol. 9, 2010, pp. 1308-1318. DOI: 10.1109/TMC.2010.76

[9] H. Huang, A. V. Savkin, M. Ding, C. Huang, Mobile robots in wireless sensor networks: A survey on tasks, Computer Networks 148, 2019, pp. 1-19. DOI: 10.1016/j.comnet.2018.10.018

[10] H. Huang, A. V. Savkin, Reactive 3D deployment of a flying robotic network for surveillance of mobile targets, Computer Networks 161, 2019, pp.172-182. DOI: 10.1016/j.comnet.2019.06.020

[11] H. Huang, A. V. Savkin, An energy efficient approach for data collection in wireless sensor networks using public transportation vehicles, AEU-Int. Journal of Electronics and Communications, 75, 2017, pp. 108-118. DOI: 10.1016/j.aeue.2017.03.012

[12] A. V. Savkin, S. C. Verma, W. Ni, Autonomous UAV 3D trajectory optimization and transmission scheduling for sensor data collection on uneven terrains, Defence Technology, vol. 30, 2023, pp. 154-160. DOI: 10.1016/j.dt.2023.03.020

[13] P. Kim, J. Park, Y. K. Cho, J. Kang, UAV-assisted autonomous mobile robot navigation for as-is 3D data collection and registration in cluttered environments, Automation in Construction 106 (2019) 102918. DOI: 10.1016/j.autcon.2019.102918

[14] H. Huang, A. V. Savkin, Viable path planning for data collection robots in a sensing field with obstacles, Computer Communications 111 (2017), pp. 84-96. DOI: 10.1016/j.comcom.2017.07.010

[15] L. E. Dubins, On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents, American Journal of mathematics, 79.3 (1957), 497-516. DOI: 10.2307/2372560

[16] T. Stützle, M. Dorigo, ACO algorithms for the traveling salesman problem, Evolutionary algorithms in engineering and computer science 4 (1999), pp. 163-183.

[17] S. Olasz-Szabó, I. Harmati, Path planning for data collection multiagent system in a sensing field with obstacles, 25th Int. Symp.

on Measurement and Control in Robotics, Houston, TX, USA, 28-30 September 2022.
DOI: 10.1109/ISMCR56534.2022.9950570

[18] A. V. Savkin, M. Hoy, Reactive and the shortest path navigation of a wheeled mobile robot in cluttered environments, Robotica 31.2 (2013), pp. 323-330.
DOI: 10.1017/S0263574712000331

[19] S. Olasz-Szabó, I. Harmati, Path planning for data collection robot in sensing field with obstacles, Acta IMEKO Vol. 11 No. 3 (2022)
DOI: 10.21014/acta_imeko.v11i3.1254

[20] P. Junjie, Wang Dingwei, An ant colony optimization algorithm for multiple travelling salesman problem, First Int. Conf. on Innovative Computing, Information and Control (ICICIC'06), Beijing, China, 30 August - 01 September 2006, Vol. 1, 2006.
DOI: 10.1109/ICICIC.2006.40

[21] J. Yang, X. Shi, M. Marchese, Y. Liang, An ant colony optimization method for generalized TSP problem, Progress in Natural Science 18.11 (2008), pp. 1417-1422.
DOI: 10.1016/j.pnsc.2008.03.028

[22] K. H. Hingrajiya, R. K. Gupta, G. S. Chandel, An approach for solving multiple travelling salesman problem using ant colony optimization, Computer Engineering and Intelligent Systems, vol. 6. No. 2, 2015, pp. 13-17. Online [Accessed 5 November 2024] https://core.ac.uk/download/pdf/234644946.pdf

[23] T. Ramadhani, G. F. Hertono, B. D. Handari, An Ant Colony Optimization algorithm for solving the fixed destination multi-depot multiple traveling salesman problem with non-random parameters, AIP Conf. Proceedings, vol. 1862, no. 1, AIP Publishing LLC, 2017, 7 pp.
DOI: 10.1063/1.4991227

[24] Li-Chih Lu, Tai-Wen Yue, Mission-oriented ant-team ACO for min–max MTSP, Applied Soft Computing 76 (2019), pp. 436-444.
DOI: 10.1016/j.asoc.2018.11.048

[25] M. Dorigo, L. M. Gambardella, Ant colonies for the traveling salesman problem, Biosystems 43 (1997) 2, pp. 73-81.
DOI: 10.1016/S0303-2647(97)01708-5

[26] G. A. Croes, A method for solving traveling-salesman problems, Operations research 6.6 (1958), pp. 791-812.
DOI: 10.1287/opre.6.6.791