ACTA IMEKO
ISSN: 2221-870X
September 2022, Volume 11, Number 3, 1 - 10

Path planning for data collection robot in sensing field with obstacles

Sára Olasz-Szabó[1], István Harmati[1]

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

ABSTRACT
Using mobile robots to collect data from wireless sensor network can reduce energy dissipation and this way improves network lifetime. Our problem is to plan paths for unicycle robots to visit a set of sensor nodes and download data on a sensing field with obstacles while minimizing the path length and the collecting time. Reconstructing the path of an intruder in a guarded area is also a possible application of this technology. During path planning we greatly emphasize the handling of obstacles. If the area contains many or large obstacles, the robots may spend long time for avoid them so this is a critical point of finding the minimal path. This paper proposes a new approach for handling obstacles during path planning. A new algorithm is developed to plan the visiting sequence of nodes taking into consideration the obstacles as well.

Section: RESEARCH PAPER

Keywords: Path planning, mobile robots, obstacle avoidance

Citation: Sára Olasz-Szabó, István Harmati, Path planning for data collection robot in sensing field with obstacles, Acta IMEKO, vol. 11, no. 3, article 5, September 2022, identifier: IMEKO-ACTA-11 (2022)-03-05

Section Editor: Zafar Taqvi, USA

Received February 26, 2022; **In final form** July 21, 2022; **Published** September 2022

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

1. INTRODUCTION

Nowadays there is a more and more common need for continuous data collection on a specified area. The simplest way for such data collection is using wireless sensor networks (WSN) [1], [2]. In most applications, a WSN consists of two parts: one data collection unit (also known as a sink or base station) and a large number of tiny sensor nodes. Typically, both sensor nodes and sink remain static after deployment. Sensor nodes, which are equipped with various sensor units, are capable of sensing the physical world and providing data to the sink through single-hop or multi-hope routing [3]. Sensors are usually powered by batteries, which cannot be replaced in some applications, e.g., battlefield surveillance.[4]. Since the data loss rate is increasing with the distance and each 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 sending is uses a lot of energy and this deteriorates network lifetime. For these reasons, the data transmission is executed by data collection robots [6], [7]. There are many applications of this technology in literature from recent years. For example, in [8] it is reviewed a range of techniques related to mobile robots in WSNs. In paper [9], considered deploying a flying robotic network to monitor mobile targets in an area of interest for a specific time period with using WSNs. In the work [10] investigated using a mobile sink, which is attached to a bus, to collect data in WSNs with nonuniform node distribution. However, the robots have limited velocity and this way the data delay is significantly increasing. Since transmitting over a short distance is more reliable than long distance, using robots improves the data collection rate. In addition, in terms of security, sending mobile sinks to collect data is more secure than transmitting via multihop communication [11]. This may be important in some military applications, as well.

In paper [12] the authors raise and solve a problem of viable path planning for data collection unicycle robots in a sensing field with obstacles. The robots must visit all sensing nodes and then return to the base station and upload the collected data. Path planning for the robots is a crucial problem since the constructed paths directly relate to the performance such as the delivery delay and energy consumption of the system. In a sensing field there are obstacles as well and the robots must not collide with them.

The data collection is carried out by unicycle Dubins-car [13], which can only move with constants velocity and bounded angular velocity, so it can move only on straight lines and turn with bounded turning radius.

For successful path planning it is necessary to determine the criteria of an adequate path. In paper [12] the authors define a viable path which is smooth, collision-free with sensor nodes/base station and obstacles, closed, and provides enough contact time with all the sensor nodes. Because of the kinematic properties of the robots the path must be smooth. Safety boundary is determined around obstacles and nodes for the sake of collision-free path. All nodes are bounded with a visiting circle with the minimum turning radius of the unicycle robot. The minimum turning radius depends on the speed of the robot and its maximum angular velocity. Moreover, all obstacle's convex hull are bounded with a safety margin, since in case of the shortest path the robot should move on 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 as long as it collects all the data from the sensor node. During path planning it is assumed that the location of all nodes and obstacles as well as the shapes of the obstacles are known. Between two objects - nodes and obstacles - there are always defined four tangents but any tangents that intersect other obstacle are removed. So when the robot arrives to a node on a tangent it starts downloading data and during it makes round trips as long as it collects all the data from the node and then it leaves the node on a tangent. So a path consists of an adequate configuration of tangents and arcs around objects at the safety distance.

The paper organized as follows. In Section 2 we summarize the basic method [12] and then Section 3 describes the proposed concepts for the path planning and also presents our new algorithms. In Section 4 we demonstrate simulation results.

2. SUMMARY OF THE SHORTEST VIABLE PATH PLANNING ALGORITHM

In paper [12] the Shortest Viable Path Planning (SVPP) algorithm was defined. The main steps of SVPP are outlined as Algorithm 1. This algorithm first computes a Σ permutation of nodes without obstacles by solving an Asymmetric Travelling Salesman Problem. For this they construct a directed graph, where the vertices are the nodes and the length of the edges are calculated as follows. The length of the edge between two vertices takes into account two aspects: the length of the valid path between their visiting circles and the length of the adjusted arc on the latter vertex. Thus, the length of the edge from s_1 to s_2 equals to the summation of the average length of tangents and the length of the adjusted arc on the visiting circle of s_2. In contrast, the length of the edge from s_2 to s_1 equals to the summation of the average length of tangents and the length of the adjusted arc on the visiting circle of s_1. With such directed graph, they use an ATSP solver [14] to calculate the permutation Σ. At this point there can be tangents that intersect obstacles in $G(V,E)$ Tangent Graph [15]. V denotes the tangent points and E denotes the tangents.

```
Algorithm 1: Shortest Viable Path Planning (SVPP)
1. Compute $\Sigma$ by solving ATSP instance based on $G(V,E)$
2. Compute $\Sigma'$ by adding those obstacles to $\Sigma$ that safety boundaries block tangents between nodes.
3. Simplify $G(V,E)$ to $G'(V',E')$ by keeping the edges and the vertices related to $\Sigma'$ and deleting others.
4. Convert $G'(V',E')$ to tree-like graph $T$.
5. Given an initial configuration, search the shortest path $P$ in $T$.
```

The second and third step of this algorithm adds the blocking obstacles to the permutation and constructs a Simplified Tangent Graph. Having Σ, $G(V,E)$ can be simplified by keeping only the tangent edges that connect succeeding visiting circles in Σ and the corresponding arc edges. When any obstacle blocks the route between any pair of visiting circles, the tangents passing the obstacle's safety boundaries are also included in the $G'(V',E')$ Simplified Tangent Graph and the algorithm inserts the obstacle to the Σ' permutation between the two nodes. One obstacle can block more than one pair of nodes. In this case the algorithm inserts the obstacle to the Σ' permutation into more than one place. The algorithm constructs a $G'(V',E')$ by keeping the edges and vertices related to the permutation of nodes and obstacles while deleting others. Obviously, $\Sigma \subseteq \Sigma'$. The new graph is called the Simplified Tangent Graph $G'(V',E')$, where $V' \subseteq V$ and $E' \subseteq E$.

The next step is converting $G'(V',E')$ to a tree-like graph T. This gives additional information about the succeeding usable tangents and arcs. From every object there are four tangents departing to the next object, the starting tangent points of these are the departure configurations, and there are four tangents arriving from the previous object, the tangent points of these are the arrival configurations. This means that every object can be transformed to 8 vertices in a tree-like graph. The path length between two objects in Σ' permutation always consist of two components. The first component is the arc around the first object from the arrival to the departure tangent point, including the additional full circles if these are necessary to download the data. The second component is the length of tangent between the two tangent points.

For the calculation of distance between the ith and $i+1$th ($i, \in [2, n'-1]$, where n' denote the number of objects in the permutation) objects, we need information about the tangent and tangent point between the $i-1$th and ith objects. One should know which tangent point will be used by the tangent on the visiting circle of the ith object in order to calculate the arc length on the visiting circle. Figure 1 illustrates this

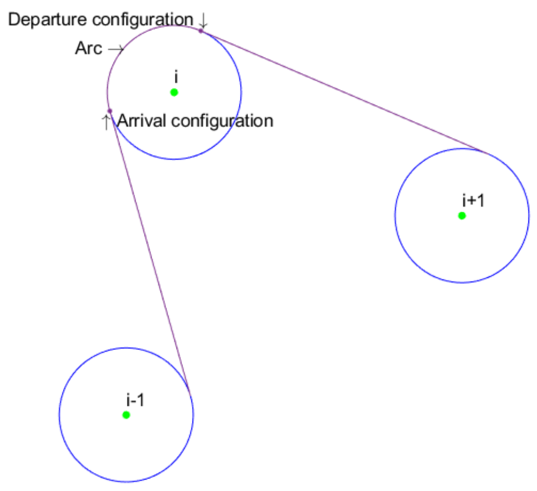


Figure 1. The distance between two objects in permutation consist of two part: arc and tangent length.

problem. The path between two objects is represented by two segments in the directed tree graph. These two parts are the arc and tangent. So in the tree-like graph the vertices are the tangent points and the edges are the arcs and the tangents. The direction of edges points to the next part of the path. During the representation of the edges one must pay attention to the heading constraints. The heading constraint refers to that the robot's heading at the beginning of an edge should be equal to that at the ending of the last edge.

The base station is the starting node, so the first element of the tree-like graph is one of the points of the base station visiting circle's. Because of closed path, the final element of the tree-like graph should be also one of the points of the base station visiting circle. Since the authors use Dubins-car, they construct the tree-like graph both for positive and negative, clockwise and anti-clockwise initial direction as well. It can be seen that from each arrival configuration of an element there are two options to reach the arrival configurations of the next element because of the heading constraint. From a given starting point the total number of paths starting and ending at this point is $2^{n'-1}$ taking into consideration that the starting and ending direction should be the same because of the continuous data collection.

In paper [12], a dynamic programming based method is used to solve the shortest path search in the tree-like graph.

3. NEW CONCEPTS OF SOLUTION

In this paper new concepts of SVPP algorithm are developed. The new algorithm based on these modifications is called Generalized-SVPP algorithm. In the following these modifications and new algorithms will be described in detail.

3.1. Constructing Tangents Graph

In paper [12] the tangents that intersect visiting circles are not included in the Tangent Graph (Assumption 1, see below). However, the robot can move collision-free on a tangent that does not intersect the circle with centre of node and radius d_{safe}. Therefore in the proposed new algorithm tangents that do not intersect the circle with d_{safe} radius around a node are available as well (Assumption 2). This way the planned path may be shorter in certain cases.

Assumption 1: The tangents that intersect visiting circles are not included in the Tangent Graph.

Assumption 2: The tangents that intersect visiting circles, but do not intersect circles with centre of a node and radius d_{safe}, are included in the Tangent Graph.

3.2. Permutation of Nodes

At this point the obstacles are not taken into account when creating the permutation of nodes. Tangents that are intersecting obstacles are allowed in this step. A graph is constructed where the vertices are the nodes and the length of the edges is the average length of tangents between the two nodes. After this there is a searching for the shortest closed cycle with all of the nodes, namely the shortest Hamilton cycle in this graph. This problem is the Travelling Salesman Problem and by solving it the Σ permutation of nodes is determined.

As it was presented in Section 2, in paper [12] the authors take into account the path length necessary to download the data and solve this problem with ATSP. The exact path length around a visiting circle cannot be determined since the actual tangents are not known at this point. This is the reason why the average length of the tangents is used.

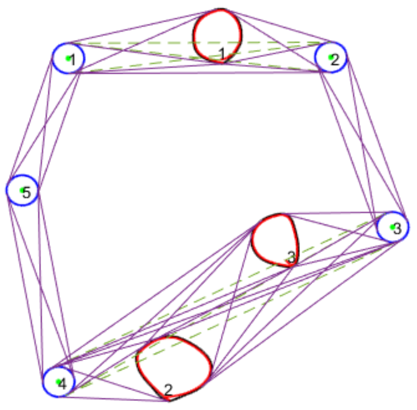


Figure 2. An example of blocked tangents are illustrated with dashed line.

3.3. New Concept of Handling Obstacles, Construction of Simplified Tangent Graph

In Algorithm 1 (SVPP) there are two types of problem of handling obstacles. First these problems are described and then the solutions for them are presented. These problems are illustrated in Figure 2.

1. For example, in Figure 2 between Node-1 and Node-2 there is only one available tangent. When Obstacle-1 is in the permutation, the available tangent between the nodes is not feasible in the shortest path planning. But when Obstacle-1 is not in the permutation, there may be no solution at all depending on the initial configurations due to heading constraints.
2. There can be more obstacles between two nodes so that these obstacles block different tangents. For instance, in Figure 2 between Node-3 and Node-4 Obstacle-2 and Obstacle-3 are blocking different tangents.

Tangent Directions: The tangent direction is called positive-negative (pn) if the robot can make round trip around the first node positive - clockwise - direction and around the succeeding second node in negative direction. The positive-positive (pp), negative-positive (np) and negative-negative (nn) directions can be defined similarly as well.

In this paper a new algorithm is proposed instead of the second and third step of Algorithm 1. Algorithm 1 creates one permutation of nodes and obstacles (Assumption 3). The basic idea of the new algorithm is to calculate more than one permutation (Assumption 4) and then using these to construct the $G'(V', E')$ Simplified Tangent Graph and then the T tree-like graph in order to get better solution.

Assumption 3: One permutation of nodes and obstacles is created. The original SVPP (Algorithm 1) uses this Assumption.

Assumption 4: More than one permutation of nodes and obstacles are created. The new Algorithm 2 created by applying this Assumption.

Instead of the second and third step of Algorithm 1 the following Algorithm 2 is used.

At first stage four copies of Σ permutation of nodes are created, then for every two nodes the blocking obstacles of all

```
Algorithm 2: Add Obstacles to Permutations
1. Create four copies of $\Sigma$ permutation
2. For every two nodes determine the blocking obstacles of all
   the four tangents and insert these obstacles to the proper
   positions in the feasible permutations according to their
   distance from the previous node.
3. Eliminate the duplicate permutations.
4. Jump to 1. and repeat the algorithm while there are
   intersecting obstacles. In this case instead of $\Sigma$ we use
   $\Sigma'$ permutations.
```

the four tangents are determined. When one of the tangents intersects an obstacle, the obstacle is inserted to the proper position in the feasible permutation determined by the tangent direction. Note that when more than one obstacle is intersected, they are inserted to the feasible permutation according to their distance from the previous node. Then the duplicate solutions are eliminated, and after this the previous algorithm is repeated as long as there are no new intersected obstacles. While repeating this algorithm, in the first step the Σ' permutations of nodes and blocking obstacles is used instead of the Σ permutation.

In first case both of the direct tangents and the edges passing the obstacle safety boundaries are also inserted into the Simplified Tangent Graph. In case one tangent blocked by more than one obstacles, all tangents and tangent points between the two nodes, between any obstacle and the two nodes, and between any two obstacles are inserted into the Simplified Tangent Graph if these tangents are not blocked. The Simplified Tangent Graph contains all of the tangents and tangent points from any permutations. Besides, it also contains all of the arcs between the tangent points.

3.4. Constructing the Tree-like Graph

The Dubins-car moves on tangents or arcs. In case of obstacles it moves on arcs between the arrival and the departure configurations and around the visiting circle while it downloads all the data from the sensor node. Because of heading constraint the tangent direction determines the direction around the next object. And the direction around the object determines the available departure tangents. There are two tangents available for a given direction for any two objects. In case of one permutation there are two departure configurations for every object, but if there is more than one permutation the count of departure configurations depend on the permutations and the Simplified Tangent Graph. In paper [12] there is only one permutation, so there are some cases when it causes problems as it was shown it the previous section. The new Algorithm 2 proposed in the present article may achieve shorter path and give more general solution, but the tree-like graph become more complex. The new algorithm can select the shorter path from more available options. In this paper Algorithm 3 and Algorithm 4 are recommended to construct the tree-like graph for cases with one and more than one permutation, respectively.

We demonstrate the transformation from Simplified Tangent Graph into tree-like graph with help of example field in Figure 3. For constructing the tree-like graph, in the first step two starting and two ending vertices are created for both the positive and negative initial direction. It is illustrated in Figure 4 a) picture.

```
Algorithm 4: Constructing the $T$ in case of more than one $\Sigma'$ permutation
1. Apply Algorithm 3 to the $\Sigma$ permutation of nodes. Add
   edges only that are the member of $G'(V',E')$.
2. Do for all $\Sigma'$ permutations:
   1.) If there are only one obstacle in the $i$th place between
       two nodes: run step 2-4 of Algorithm 3 for the
       $\sigma_{i-1}, \sigma_i, \sigma_{i+1}$ object. Only if the edges are in the
       $G'(V',E')$.
   2.) If there are more than one obstacles between two
       nodes. Let denote the obstacles by $\partial O = \{\partial o_1, \dots, \partial o_j\}$:
       do step 1.) for all $\partial o_i \in \partial O$ obstacle in the given
       permutation and for the nodes before and after the
       obstacle. Run step 2-4 of Algorithm 3 for all pair of
       obstacles $\partial o_i \in \partial O$. Run it in that case also if these are
       not subsequent elements in the permutation. Naturally
       only in case when the vertices and edges are in
       $G'(V',E')$.
```

```
Algorithm 3: Constructing the $T$ in case of one $\Sigma'$ permutation
1. For both negative and positive direction add starting and
   ending point as vertices to $T$.
2. According to $\Sigma'$ add 4 arrival and 4 departure tangent points
   for all objects to the $T$
3. Determine all arc length between all possible arrival and
   departure configurations taking into consideration the
   heading constraint and the possible different arc length
   calculation methods of the nodes and obstacles. Add arc
   lengths as the length of the edges to the $T$ between the
   corresponding vertices.
4. Around the visiting circle of the base station determine arc
   length between the starting point and the arrival and
   departure configurations. Add this as edge length to $T$ to the
   proper position.
5. According to $\Sigma'$ add all tangents between all consecutive
   objects to the $T$ taking into consideration the heading
   constraint.
```

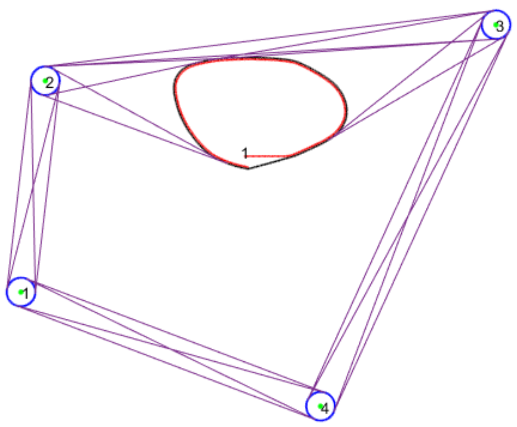
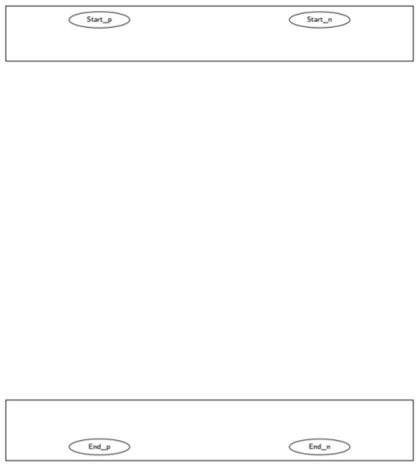
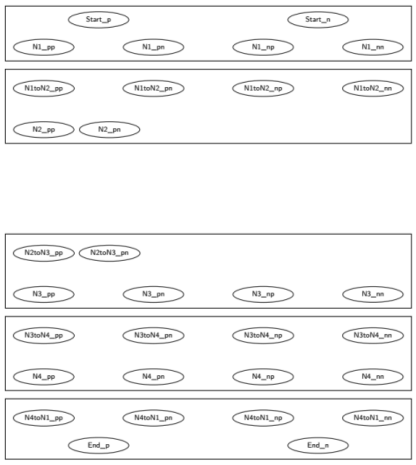
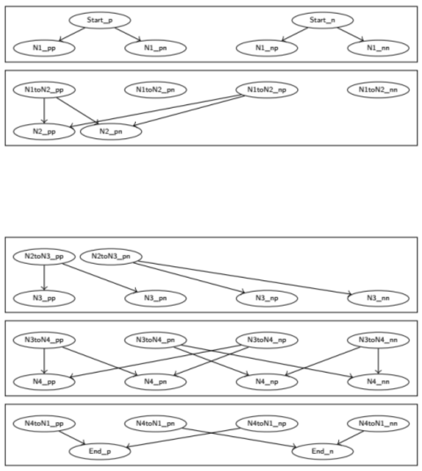
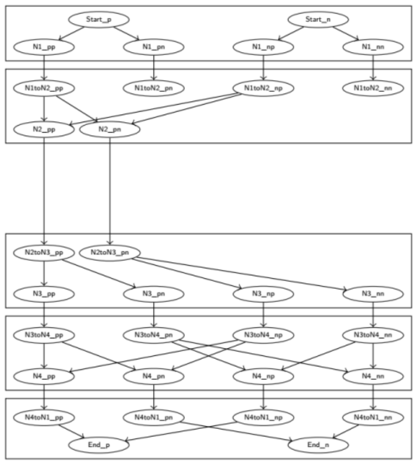


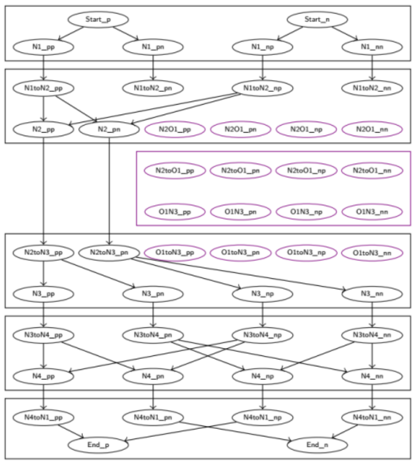
Figure 3. An example field of tree-like graph construction with $\Sigma = \{C_1, C_2, C_3, C_4\}$ permutation of nodes

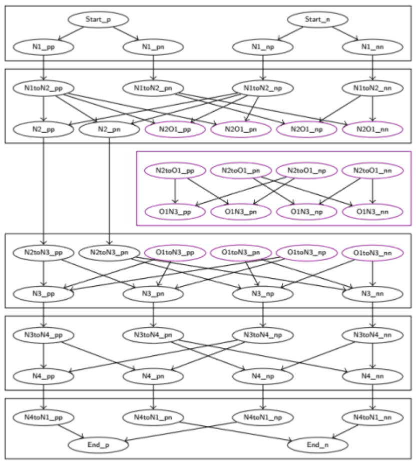

a) Add starting and ending points as vertices


b) Add tangent points of nodes as vertices


c) Add arc length around the nodes between the arrival and departure configurations as edges


d) Add tangent length between the departure and arrival configurations as edges


e) Add tangent points that tangents between the obstacles and nodes in Simplified Tangents Graph as vertices


f) Add arc length around the obstacles and the connecting nodes as edges

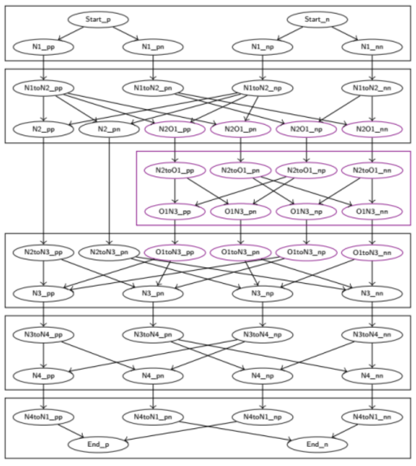

g) Add tangents between the obstacles and nodes as edges

Figure 4. An example of tree-like graph construction from Figure 3 using Algorithm 4. Obstacles are denoted by purple. The vertices of the same rectangle denote the tangent points of the same object

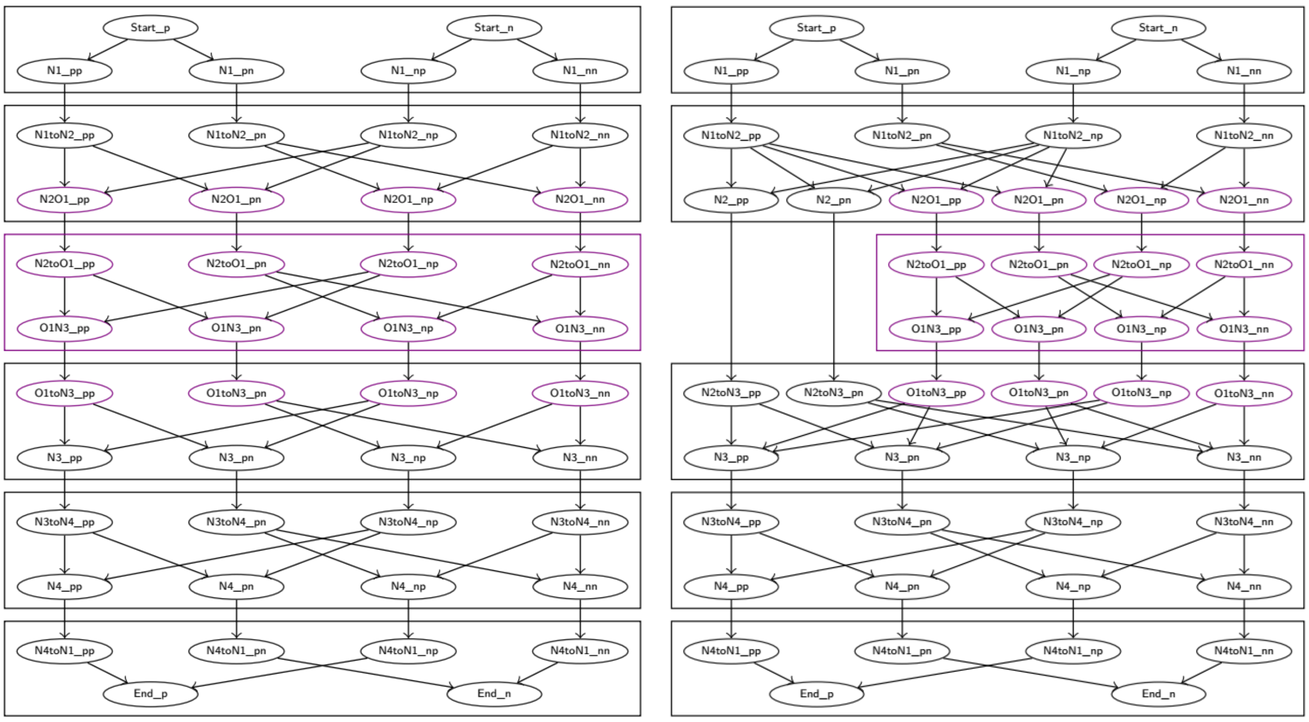


Figure 5. An example of tree-like graph construction from Figure 3 using Algorithm 3 and Algorithm 4. Obstacles are denoted by purple.

After this the nodes are inserted into the graph, according to permutation Σ. Using $G'(V', E')$ vertices are created from the tangent points (Figure 4 b) picture). The next step is to determine the edges between the vertices. The length of edges between the starting point and the departure configurations of the visiting circle of the base node is equal to the arc length between the starting point and the departure configurations.

The next step is to determine the arc length between the arrival and departure configurations around the node taking into account the heading constraint. In next step edges are added to T between the ending vertices and the arrival configurations of the base node's visiting circle (Figure 4 c) picture). Finally, between the departure configuration and the arrival configurations of the next node, the length of the edges are the length of the tangents with the proper direction (Figure 4 c) picture). This step should be done for all nodes in the order given by Σ permutation.

Next step is to add obstacles to the tree-like graph according to Σ' permutation and $G'(V', E')$ Simplified Tangent Graph. In case of Figure 3 Obstacle-1 blocks tangents between Node-2 and Node-3. First those departure tangent points are added as vertices of the visiting circle of Node-2 which are also on a tangent of Obstacle-1. In addition, the arrival configuration of the obstacle coming from Node-2 and departure configurations coming from the obstacle to the Node-3 are added as vertices to the tree-like graph. Finally, the arrival configurations of the Node-3 are added to tree-like graph (Figure 4 e) picture). Adding edges is similar as it was previously (Figure 4 f)-g) picture).

It might occur that there is more than one obstacle between two nodes. In this case all existing tangent points and tangents between the previous and next nodes to/from all obstacles are added as vertices and edges to the tree-like graph. The tangents and tangent points between all pair of obstacles in the proper edge direction are added as vertices and edges.

In paper [12] the SVPP algorithm handles obstacles in the same way as nodes when these are added to the tree-like graph. The authors iterate step by step on permutation Σ' by adding all tangent points to the tree-like graph as vertices. Then they add arcs and tangents to the tree-like graph as edges taking into consideration the heading constraint. The tree-like graph constructed from the Figure 3 sensing field can be seen on Figure 5 using both Algorithm 3 and Algorithm 4. During the construction of the tree-like graph according to Algorithm 4 first vertices were created from nodes denoted by black then the edges were added between them. Then the tangent points of obstacles were added as vertices denoted by purple and the associated edges were also added. The different objects are separated with rectangles in the figure. It can be seen that in Figure 5 on the second picture there are direct tangents between Node-2 and Node-3, namely edge between vertices N2_pp and N2toN3_pp, but on the first picture of Figure 5 there are only paths using edges that passing Obstacle-1 (this is because of Assumption 1 and Assumption 3).

Theorem Using our new Assumption 2 and Assumption 4 the planned path always better or equal to the path that using the original Assumption 1 and Assumption 3.

Proof The Simplified Tangents Graph in case of Assumption 2 or 4 always contains all edges and vertices from Simplified Tangents Graph using Assumption 1 and 3. Therefore the tree-like graph using Assumption 2 and 4 always contains the tree-like graph using Assumption 1 and 3 as well. So the tree-like graph using Assumption 2 and 4 contains all paths that the tree-like graph using Assumption 1 and 3 and possibly even more paths. Therefore, the shortest path in the tree-like graph using Assumption 2 and 4 always shorter or equal to the shortest path of the tree-like graph using Assumption 1 and 3. □

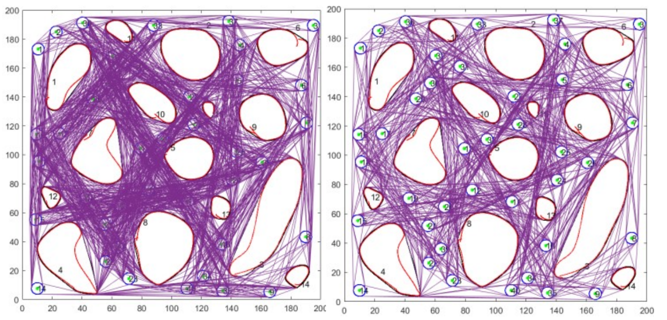


Figure 6. The difference between the $G(V,E)$ Tangent Graphs using Assumption 1 and Assumption 2.

3.5. Searching the Shortest Path in a Tree-like Graph

After the tree-like graph was created, it was searched for the shortest path. There are many different way to find the shortest path. Here in this paper the Dijkstra algorithm [16] were applied. The searching for the shortest path was carried out in both positive and negative initial direction and then the shorter was selected.

3.6. Complexity of G-SVPP algorithm

To end this section, we analyse the time complexity of G-SVPP algorithm both of Assumption 1 and 3 or Assumption 2 and 4 cases. First step is solving TSP with use of Miller-Tucker-Zemlin formulation [17] with $\mathcal{O}(n^2+n)$ computation effort. In Step 2, we check n pairs of visiting circles to see whether they are blocked by any boundary of convex hull. In each checking, check all the m obstacles and then the time complexity is $\mathcal{O}(nm)$. In Step 3, we do a constant number of operations to each permutation each element in Σ', then the time complexity of the simplifying procedure is $\sum_{i=1}^{n_{\Sigma'}}\mathcal{O}(n'_i)$, where n'_i denote the length of the ith permutation of Σ' for each permutation and $n_{\Sigma'}$ denote the number of permutations of Σ'. Converting $G'(V',E')$ to T costs $\mathcal{O}(1)$ in Step 4. The shortest path searching of the T is implemented Dijkstra algorithm [16]. The computation effort of Dijkstra algorithm is $\mathcal{O}(|E'|+|V'|^2)=\mathcal{O}(|V'|^2)$ so that depend on the number of vertices of T tree like graph. In case Assumption 1 and 3 the T tree like graph contains maximum $8n'+4$ vertices, since there are four arrival and four departure configurations all objects, and two starting and two ending vertices of the base station. In case Assumption 2 and 4 the tree-like graph in worst case contains $\sum_{i=1}^{n_{\Sigma'}}8n'_i+4$ vertices. Therefore, the worst case computation effort in case Assumption 2 and 4 is $n_{\Sigma'}{}^2$ times larger than the maximum computation error in case Assumption 1 and 3. However, in most cases the Σ' permutations differ from each other in only a few elements so the computation error is more smaller than in the worst case.

4. SIMULATION RESULTS

In the present paper a $200\ \text{m}\times 200\ \text{m}$ virtual field was simulated with 40 nodes, of which one is the base station and the other 39 are sensor nodes. The base node is Node-1. Each sensor node stores $g=0.5\ \text{MB}$ data and to the base node $g_B=(n-1)\ 0.5\ \text{MB}=19.5\ \text{MB}$ collected data is uploaded by the robot for further analysis. The data transmission rate at the visiting circle is $r=250\ \text{kB/s}$. In a sensing field there are 15 obstacles as well. The robot speed is $v=4\ \text{m/s}$ and the maximal angular velocity is $|u_M|\le 1\ \text{rad/s}$, therefore the minimal turning radius and also the visiting circle's radius is $R_{min}=v/u_M=4\ \text{m}$ the robot must move at least $d_{safe}=0.5\ \text{m}$ distance from an object in order to avoid to collision.

The next step is to construct the $G(V,E)$ Tangent Graph, for this the tangents and the tangent points between the objects are determined and then the edges are deleted according to Assumption 1 or Assumption 2. The difference between the two assumptions can be seen in Figure 6. It can be seen in Figure 6 that the proposed Assumption 2 produces more tangents, since this allows tangents that intersect visiting circle but not intersect circle with centre of node and radius d_{safe}. Therefore, increases the number of possible paths so it is feasible shorter path planning. But at the same time, it requires more computations as well.

The next step is to determine the Σ permutation of nodes and then construct the Σ'permutation or permutations with obstacles depending on Assumption 3 or Assumption 4. Using Σ' the $G'(V',E')$ Simplified Tangent Graph is constructed. The $G'(V',E')$ using Assumption 3 and Assumption 4 can be seen in Figure 7 and Figure 8. In Figure 7 the Tangents Graph

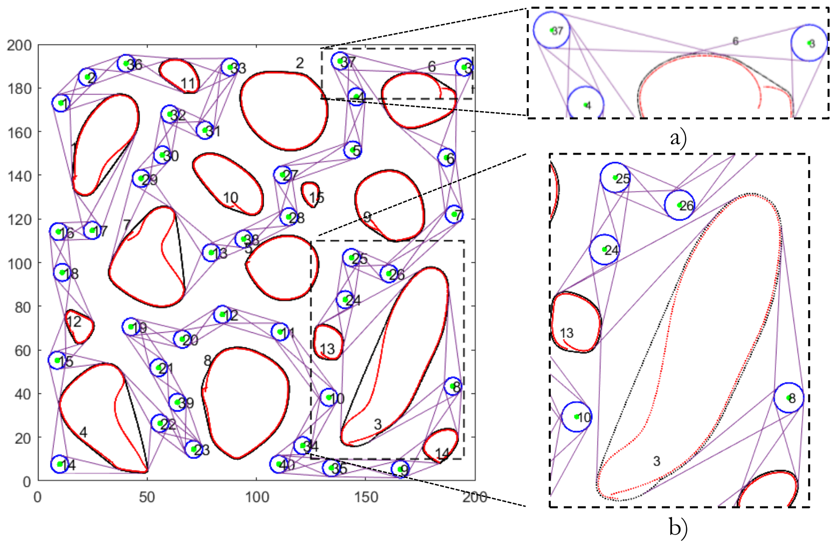


Figure 7. $G'(V',E')$ Simplified Tangent Graph using Assumption 1 and Assumption 3 .

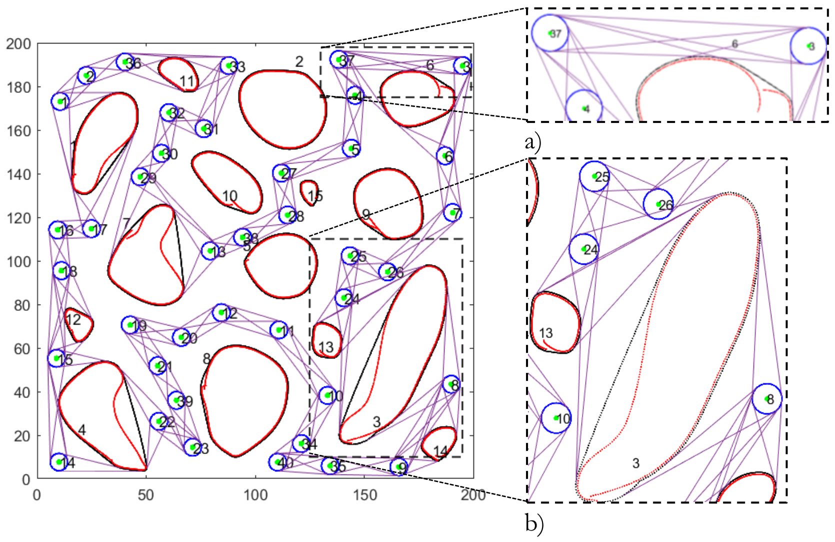


Figure 8. $G'(V',E')$ Simplified Tangent Graph using Assumption 2 and Assumption 4 .

constructed by Assumption 1 is simplified by applying Assumption 3. Similarly, in Figure 8 it can be seen the $G'(V',E')$ applying Assumption 2 and Assumption 4. In Figure 8 it can be seen that using the Assumption 2 and Assumption 4 proposed in this article, the tangents that intersect visiting circles are available and more permutations can be constructed so there are more available tangents in a $G'(V',E')$ and therefore in the tree-like graph as well. In this way, there are more possible paths and therefore the algorithm may plan shorter path. In this case the constructed tree-like graph is more complex than in case of Assumption 3. When we use Assumption 1 and Assumption 3 there can be at most 4 arrival and 4 departure tangents for every object in the order of the permutation. Usually there are four-four arrival and departure tangents, but for example in Figure 7, between Obstacle-13 and Obstacle-3 there are just 3 available tangents. In Figure 8 it can be seen the case of Assumption 2 and Assumption 4. Since in the second picture the fourth tangent intersects the safety circle of Node-10 it is also not available in the Simplified Tangent Graph. In Figure 8 there are direct tangents between Node-24 and Obstacle-3 and one of them intersects the visiting circle of Node-26, but in Figure 7 the robot must visit Obstacle-13 first. In Figure 8 there are three available tangents between the visiting circles of Node-37 and Node-3, but in Figure 7 (Assumptions 1 and 3) the robot can only move on tangents that passing Obstacle-6.

The next step is to construct the T tree-like graphs for both cases Assumption 1 and 3 or Assumption 2 and 4. Finally, in the tree-like graph a search for the shortest path is carried out both for positive and negative initial direction as well. In Figure 9 and Figure 10 it can be seen that the planned path using the new Assumptions 2 and 4 proposed in the present article is shorter than in the original case. In this example, the planned path with positive and negative initial direction only differs in the tangents that are belonging to the base station. Using Assumptions 2 and 4 the planned path between the Node-24 and Obstacle-3 use tangent that intersect the visiting circle of Node-26, therefore a shorter path can be achieved using the new Assumptions of the present article. Applying Assumptions 1 and 3 between Node-3 and Node-37, the planned path uses tangents that pass the Obstacle-6. On the contrary, if Assumptions 2 and 4 are applied, the planned path uses direct

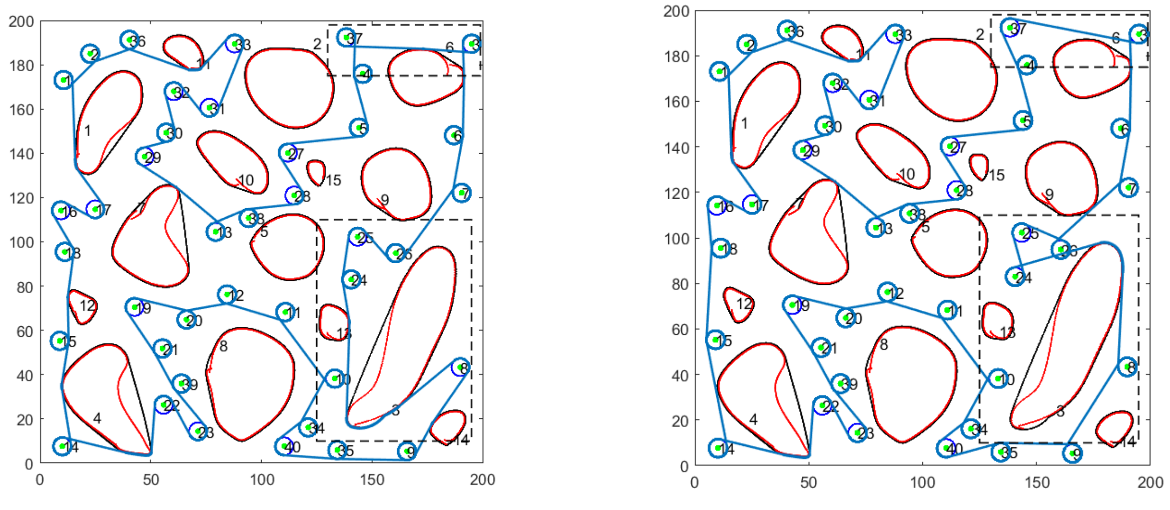


Figure 9. The resulted shortest path with negative initial direction of the T tree-like graph both for the cases using Assumption 1 and Assumption 3 or Assumption 2 and Assumption 4. The length of the planned path are $2341.94m$ and $2290.85m$, respectively.

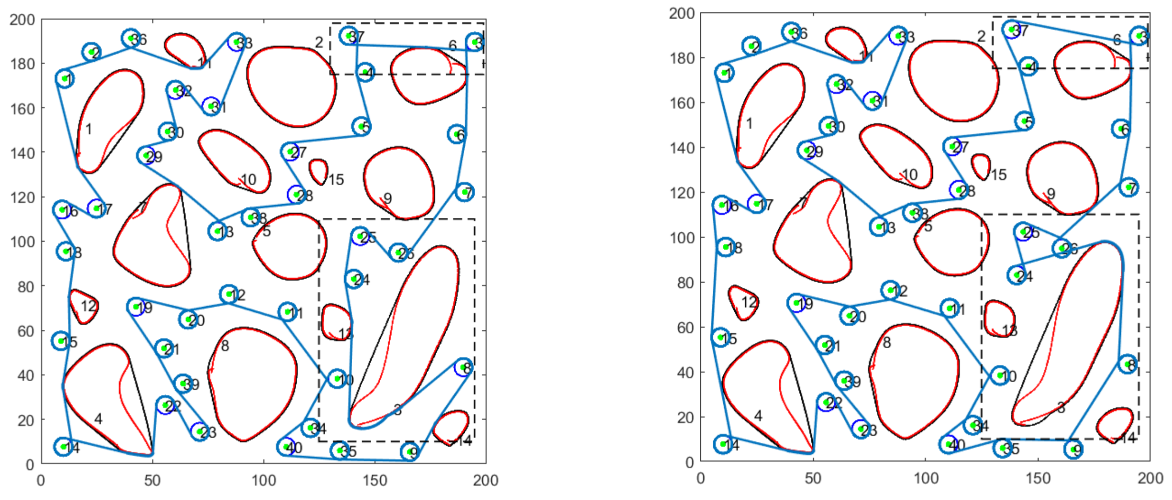


Figure 10 The resulted shortest path with positive initial direction of the T tree-like graph both for the cases using Assumption 1 and Assumption 3 or Assumption 2 and Assumption 4. The length of the planned path are $2337.64m$ and $2286.55m$.

tangents between the two nodes and this way the determined arc length is shorter.

Each sensor node data downloading requires $l = g\frac{v}{r} = 8\ \mathrm{m}$ arc length. The visiting circles circumference equal to $K = 25.12\ \mathrm{m}$. Therefore the robot must take at least approximately one third of the visiting circle's circumference to have enough time for data transmission. It can be seen that the algorithm preferably chooses tangents between Node-31, Node-32 and Node-33 in such a way that the robot is not required to make extra round trips around these nodes.

As a test, the path planning was run for ten different virtual sensing fields. The length of the planned paths can be seen on Table 1. The solutions using Assumption 1 and 3 are compared with the solutions using Assumption 2 and 4. As it was proven in the Theorem, using the new assumptions proposed in the present article the planned path always better or equal to the path using the original assumptions presented in [6]. The number of vertices on a T tree-like graph also represented in Table 1. The maximum difference between the number of vertices of Assumption 2 and 4 or Assumption 1 or 3 is 54, in test field 5. In this case the computation effort in case of Assumptions 2 and 4 is increased by $27\ \%$ compared to Assumptions 1 and 3. At the same time the planned path in case of Assumptions 2 and 4 is $92\ \mathrm{m}$ shorter than Assumptions 1 and 3. That means, $t = \frac{92\ \mathrm{m}}{4\ \mathrm{m/s}} = 23\ \mathrm{s}$ time for each period, therefore the robot can make 6 extra round trip for a day.

5. CONCLUSIONS

In the present paper a path planning algorithm was developed for unicycle robots between sensor nodes. The task is to collect all the data from the sensor nodes and then upload it to the base node. To increase the effectiveness of data collection, the length and the duration of the trip should be minimized, while also maintaining a collision-free path around the nodes and obstacles. New algorithms were developed for handling the obstacles and new assumptions were applied to reach a collision-free state. A new algorithm for tree-like graph generation was also developed, where the search of the shortest viable path will take place finally. The present paper finished with the detailed presentation of simulation results. The preparation of the field and the steps of path planning algorithm were illustrated with figures. The present article also compared the results of the new algorithm with the results of a previous research. In conclusion, the new algorithm presented in this article proved to achieve shorter paths then the earlier algorithms.

Table 1 The length of the planned path for different virtual sensing fields using both positive and negative starting directions. $|V'|$ denote the number of vertices of the tree-like graph on which it depends the computation error.

Test field	Assumption 1 and 3			Assumption 2 and 4		
	Positive	Negative	$\lvert V'\rvert$	Positive	Negative	$\lvert V'\rvert$
1	2326.23m	2233.61m	372	2254.71m	2222.65m	386
2	2393.66m	2387.67m	418	2332.58m	2326.59m	462
3	2305.82m	2313.80m	404	2270.74m	2278.72m	444
4	2372.37m	2359.42m	436	2363.87m	2350.92m	466
5	2413.25m	2403.51m	412	2321.40m	2311.65m	466
6	2294.77m	2288.91m	402	2193.65m	2189.18m	452
7	2335.53m	2317.82m	396	2312.25m	2294.55m	426
8	2330.93m	2324.78m	378	2330.51m	2324.36m	386
9	2247.02m	2262.38m	372	2238.79m	2254.14m	386
10	2301.88m	2311.12m	360	2301.88m	2311.12m	368

ACKNOWLEDGEMENT

The research reported in this paper and carried out at the Budapest University of Technology and Economics was supported by the “TKP2020, Institutional Excellence Program” of the National Research Development and Innovation Office in the field of Artificial Intelligence (BME IE-MI-SC TKP2020). The research was supported by the EFOP-3.6.2-16-2016- 00014 project - financed by the Ministry of Human Capacities of Hungary.

The research reported in this paper is part of project no. BME-NVA-02, implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021 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, (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. Conference on Mobile Systems, Applications, and Services, ACM, Boston, Massachusetts, USA, (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, IEEE Wireless Communications, vol. 11, no. 6, 2004, pp. 6-28.
DOI: 10.1016/j.proeng.2012.06.320

[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] 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, 2006, pp. 52-59.
DOI: 10.1145/1164783.1164793

[7] 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

[8] Hailong Huang, Andrey V. Savkin, Ming Ding, Chao 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

[9] Huang, Hailong, Andrey 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

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

[11] 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, pp. 507-524.
DOI: 10.1109/COMST.2015.2388779

[12] Hailong Huang, Andrey 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

[13] 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, vol. 79 No. 3,1957, pp. 497–516.
DOI: 10.2307/2372560

[14] A. M. Frieze, G. Galbiati, F. Maffioli, On the worst-case performance of some algorithms for the asymmetric traveling salesman problem, Networks, vol. 12 no. 1, 1982, pp. 23–39.
DOI: 10.1002/net.3230120103

[15] A. V. Savkin, M. Hoy, Reactive and the shortest path navigation of a wheeled mobile robot in cluttered environments, Robotica. vol. 31 issue 2,2013, pp. 323-330.
DOI: 10.1017/S0263574712000331

[16] E. W. Dijkstra, A Note on Two Problems in Connexion with Graphs, Numerische Mathematik, vol. 1, number 1, 1959, pp. 269-271.
DOI: 10.1007/bf01386390

[17] C. E. Miller, A. W. Tucker, R. A. Zemlin, Integer Programming Formulation of Traveling Salesman Problems. J. ACM 7, 4, October 1960, pp. 326-329.
DOI: 10.1145/321043.321046