BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **25**, No 3
Sofia • 2025 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2025-0029

Advanced Trajectory Planning for Unmanned Aerial Vehicles in the Context of Data Collection from Spatially Distributed Wireless Sensor Networks

Tulkin Matkurbanov[1], *Akhmet Utegenov*[2], *Mengliyev Davlatyor*[3], *Dilshod Matkurbonov*[2]

[1]*Urgench State University named after Abu Rayhan Biruni Urgench, 222100 Uzbekistan*
[2]*Tashkent University of Information Technologies Named after Muhammad al-Khwarizmi, Tashkent, 100000 Uzbekistan*
[3]*Cyber University, Nurafshan, 111500 Uzbekistan*
E-mails: tulkinmatkurbanov2020@gmail.com akhmetutegenov0402@gmail.com shogunuz@gmail.com d.m.matkurbonov@gmail.com

Abstract: *Wireless Sensor Networks* (*WSNs*) *are extensively used for monitoring large-scale areas with sensors having different coverage zones. Unmanned Aerial Vehicles* (*UAVs*) *are deployed to efficiently collect data from these distributed nodes. The effectiveness of this process depends on optimizing the UAV's flight path, formulated as the Close Enough Traveling Salesman Problem* (*CETSP*)*, known to be NP-hard. This study presents hybrid methods that integrate heuristic algorithms with geometric strategies to solve the CETSP efficiently. The main contribution lies in proposing efficient, fast-executing, and easily programmable approaches that follow a structured process: identifying new target points when zones intersect, determining near-optimal visiting sequences with heuristic algorithms, and applying iterative geometric refinements to shorten the route. A total of 92 hybrid algorithms based on seven distinct approaches are evaluated using four performance metrics. Experimental results demonstrate the high efficiency of the proposed methods and their strong potential for real-world UAV-assisted data collection tasks in wireless sensor networks.*

Keywords: *Unmanned aerial vehicle, Traveling salesman problem, Traveling salesman problem with neighborhoods, Close enough traveling salesman problem, Wireless sensor networks.*

1. Introduction

In recent years, the field of Unmanned Aerial Vehicles (UAVs) technology has made significant progress, leading to a notable increase in their application across various domains. In many areas such as monitoring, security, agriculture, and the military, UAVs occupy a unique and indispensable position [1]. These applications typically involve the use of Wireless Sensor Networks (WSNs) for data collection

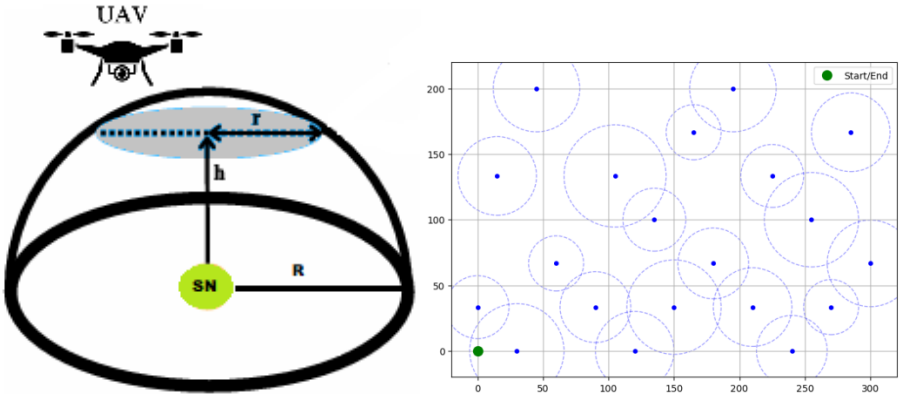
and enhanced communication capabilities. WSNs currently serve as a fundamental component of the Internet of Things (IoT), providing crucial infrastructure for real-time sensing and information exchange [2], [30].

To overcome the limitations of traditional WSNs, extensive research is being conducted on the integration of **UAVs and WSNs** to enable long-range communication across various applications. In monitoring systems, UAVs are considered **mobile data collectors**, capable of gathering information from individual sensors or multiple sensors simultaneously. Fig. 1a shows an idealized geometric representation of the sensor coverage area used in WSNs [3]. Despite their advantages, determining effective and optimized UAV flight trajectories remains a challenging research problem.

When sensors are deployed over a large area and it is not feasible to establish a permanent communication network, UAV can be used for data collection by flying through the spherical radio coverage zones surrounding the sensors [4, 31].

Such problems are formulated as the Traveling Salesman Problem with Neighborhoods (TSPN), which is a generalization of the classical Traveling Salesman Problem (TSP). In the classical TSP, the objective is to find the shortest closed path that visits each given point exactly once and returns to the starting location. In contrast, the TSPN extends this by introducing "neighborhoods" – regions (typically circles) that the salesman must enter, without necessarily reaching a specific point inside them [5]. The goal is to compute the shortest trajectory that visits each region at least once [6]. There are several extended versions of this model, each adapted to specific real-world applications. When the regions are circles of varying radii, the problem is referred to as the CETSP – the most common special case of TSPN. In CETSP, it is sufficient for the salesman to enter or touch each circular region reaching the exact center is not required. The CETSP was first introduced by Gulczynski, Heath and Price [7]. As CETSP is a relatively recent modification of the TSP, the number of exact algorithms available for solving it remains limited. Due to the low performance of existing exact methods, metaheuristic and approximation algorithms are predominantly used in practice.

UAV are widely utilized in the civilian sector due to their high mobility, low operational costs, and ability to hover in place. Their applications range from environmental monitoring and search-and-rescue operations to delivery services, wireless communication, and precision agriculture [8]. A key advantage of UAVs lies in their independence from terrestrial infrastructure, enabling them to efficiently cover dispersed areas. To maximize operational efficiency, however, it is essential to plan and coordinate the UAV's flight trajectory in advance. In this context, the CETSP emerges as a highly relevant model, as it effectively captures practical constraints encountered during UAV deployment. This work addresses the construction of an efficient UAV trajectory that starts at the origin point (0, 0), traverses a set of zones represented by circles (each associated with a target or client), and returns to the starting location. Fig. 1b illustrates the coverage areas of sensors deployed over a 300×200 unit rectangular region.



(a) (b)

Fig 1. General representation of the sensor coverage zones: geometric representation of sensor coverage areas in WSN (a); two-dimensional visualization of sensor coverage zones (b)

Articles [33] and [37] present comprehensive survey studies on UAV–WSN integration. These works systematically analyze clustering, route optimization, energy efficiency, and security issues, highlighting the strengths and limitations of existing approaches. Articles [34] and [38] examine UAV-assisted data collection within the framework of software-defined wireless sensor networks. The authors propose approaches aimed at improving energy efficiency and transmission reliability through dynamic trajectory control: one based on deep reinforcement learning and the other employing fuzzy-based path planning. The three studies [32, 35, 36] focus on the problem of UAV path planning. Article [36] provides a systematic review of existing algorithms, outlining their advantages and limitations. In [35], the authors propose the “Hyperion” algorithm, which ensures UAV return to a charging station while accounting for energy constraints. Article [32] addresses multi-UAV data collection by optimizing flight routes through a genetic algorithm. A common feature of these works is their strong emphasis on improving path planning efficiency; however, a key limitation is the lack of sufficient real-world experimental validation.

Prior research has demonstrated that the CETSP framework can be effectively employed in UAV trajectory planning tasks. The following main approaches for CETSP have been observed in the literature: discretization-based heuristic approaches [9], evolutionary and geometric heuristic approaches [10-12], approximate models [13]. These approaches show that the quality of the resulting UAV trajectory is influenced by several factors: the precision of neighborhood discretization, the strategy for selecting representative points, the number and spatial density of sensor zones, algorithmic flexibility, and overall computational efficiency.

Each approach demonstrates specific advantages and limitations; therefore, for practical applications, the choice of algorithm must reflect the characteristics of a given UAV mission. Unlike earlier studies, there is a need for approaches that guarantee stable route quality under varying key parameters and allow systematic comparison with existing methods. The task of constructing efficient UAV

trajectories for data collection from sensor nodes – each represented by a coverage zone – remains insufficiently addressed and requires new solutions. Consequently, effective route planning for large-scale UAV monitoring continues to be a relevant and open research challenge. This article focuses on constructing efficient UAV routes for the CETSP in the context of data collection from sensor networks distributed across vast territories. The main contribution of this work lies in the proposal of hybrid methods that integrate heuristic algorithms with geometric strategies to efficiently solve the CETSP, along with their comparative analysis. The approaches are designed to be efficient, fast-executing, and easily programmable, and they follow a clear research roadmap. Specifically: 1 – new target points are identified in cases where coverage zones intersect; 2 – a near-optimal visiting sequence of the zones is determined using heuristic algorithms; 3 – an iterative geometric path-shortening procedure is applied to reduce the total route length; and 4 – an optimization method is used to further refine the trajectory. All computational procedures and simulations were implemented in Python, enabling reproducible and extensible experimentation with the proposed algorithms.

In this work, a three-level terminology is adopted in order to ensure consistency and avoid ambiguity. An *approach* denotes the general conceptual strategy for solving the CETSP and its UAV trajectory planning variants (e.g., “Three-dot line”, “Overlap Three-Point”, “Within_r”). A *method* refers to a specific principle or procedure applied within an approach (e.g., overlap, GEO-6, 2-OPT, Optimal or Two-Step Insertion). An *algorithm* represents the explicit computational sequence that realizes a method, or a combination of methods, in practice.

To provide a clear overview, the research process in this work follows a structured roadmap. First, the CETSP problem is formulated for UAV-based data collection in large-scale wireless sensor networks. Second, seven groups of hybrid algorithms are designed by integrating heuristic optimization methods with geometric heuristics. Third, the algorithms are implemented in Python to ensure reproducibility and extensibility. Fourth, extensive simulations are conducted on datasets containing 100 sensor zones under identical computational conditions. Fifth, the performance of the proposed algorithms is evaluated using four key metrics: total path length, CPU time, number of maneuver points, and relative deviation from sensor centers. Finally, a comparative analysis of 92 hybrid algorithms is presented, highlighting their strengths and limitations depending on sensor distribution and coverage radius.

2. Modeling and methodology

2.1. Modeling

Over a large area, N sensors are deployed, each of which must transmit its measurements or data. These nodes are distributed across a predefined extensive territory. Since each sensor can only transmit data over a limited distance, they are unable to send signals directly to a base or control station. Therefore, it is assumed that a UAV is used to collect the data in this scenario. Based on this, a set of n circles of varying radii is considered, located in the Euclidean plane (see Fig. 1b).

Each circle i is defined by the coordinates (x_i, y_i, r_i), where (x_i, y_i) are the coordinates of the circle's center, and r_i is its radius. The circles may intersect with one another. The objective is to construct a trajectory of minimal total length that passes through all the given circles. The trajectory consists of linear segments connecting selected points within these circles. There are no constraints on the radii or mutual positions of the circles. The problem consists in selecting a single point within each circle and constructing the shortest possible trajectory that passes through all the selected points. In some cases, a single point may belong to multiple circles.

The UAV starts its movement from an initial point p_0 and, sequentially visiting N nodes, eventually returns to p_0. Each node is defined by coordinates $v_n \in \mathbb{R}^2$, where $n = 1, 2, ..., N$.

The communication range of each node is represented as a circular zone (disk) with radius r_i,

$$V_n = \{x \in \mathbb{R}^2 \vee \|x - v_n\|_2 \leq r_i\}.$$

If the communication zones of two nodes partially overlap, these nodes are considered neighbors. This situation is formally described as follows:

$$N(V_n) = \{j \in \{1, \dots, N\} \vee V_j \cap V_n \neq \emptyset\}.$$

The following assumptions are also made: the initial UAV position u_0 and the final position u_{n+1} are not located within the communication zone of any node:

$$u_0, u_{N+1} \notin n = 1\mathrm{NV}_n,$$

if the initial position u_0 or the final position u_{n+1} falls inside the communication zone of any node, then:

$$u_0, u_{N+1} \in n = 1\mathrm{NV}_n,$$

if one node lies within the communication zone of another node, this is interpreted as an overlap of communication zones (interference or spatial proximity). If node v_i lies within the communication zone of node v_j, it is expressed as

$$v_i \in V_j, \text{ i.e., } \left\|v_i - v_j\right\|_2 \leq r_j.$$

The set of overlapping zones between nodes is defined as

$$N(V_n) = \{j \in \{1, \dots, N\}, j \neq n \vee V_j \cap V_n \neq \emptyset\}.$$

If $v_i \in V_j$, this implies that $V_i \subseteq V_j$ or at least $V_i \cap V_j \neq \emptyset$. This means node v_i is located very close to node v_j. As a result, the UAV can potentially establish connections with two or more nodes while positioned within a single region. This approach improves efficiency, as the UAV can collect data from multiple nodes without requiring additional movement.

Problem statement

Each circle is defined as

$$C_i = \{y_i = (u_i, v_i) | (u_i - a_i)^2 + (v_i - b_i)^2 \leq r_i\}.$$

The primary goal of the model is to minimize the total route length constructed through the selected points located inside the given circles:

$$\min \sum_{i \in C} \sum_{j \in C} p_{ij} \times d_{ij}.$$

Here: p_{ij} = 1 if circle j is visited immediately after circle i; d_{ij} is the distance between the selected points inside circles i and j.

The constraints are five.

1) Each circle must be visited exactly once:

$$\sum_{j\in C} p_{ij} = 1,\ \sum_{j\in C} p_{ji} = 1\ \ \forall i \in C.$$

2) Subtour elimination constraint (Miller–Tucker–Zemlin formulation):

$$\mathrm{seq}_i - \mathrm{seq}_j + N \times p_{ij} \le N - 1\ \ \forall i \ne j,\ i,\ j \ne 1.$$

3) Selected point must lie within the corresponding circle:

$$r_i^2 - (u_i - a_i)^2 - (v_i - b_i)^2 \ge 0\ \ \forall i \in C.$$

4) Distances between selected points must not take invalid (underestimated) values, i.e., they must be no less than the Euclidean distance:

$$d_{ij}^2 \ge \left(u_i - u_j\right)^2 + \left(v_i - v_j\right)^2\ \ \forall i \ne j.$$

5) Route-following variables must be binary:

$$p_{ij} \in \{0, 1\}\ \ \forall i,\ j \in C.$$

Since the model expresses distance in squared form (d_{ij}^2), it is nonlinear. However, this constraint can be linearized as follows:

$$d_{ij}^2 + M \times (1 - p_{ij}) \ge \left(u_i - u_j\right)^2 + \left(v_i - v_j\right)^2,$$

$$p_{ij} \ge 0.$$

Here, *M* is a sufficiently large number to ensure correct behavior of the model constraints.

This model represents a generalization of the classical TSP, incorporating circular reachability constraints. For each "point", any location inside the corresponding circle is allowed. This may result in overlapping selected points. Despite its nonlinear nature, the model allows linearization, enabling practical solutions.

2.2. Methodology

All proposed approaches are structured within a unified four-stage methodological framework. First, an initial route is constructed by solving the TSP over sensor zone centers or merged representative points. Second, geometric refinement is applied to determine UAV entry points. Third, the trajectory is optimized through classical TSP heuristics. Finally, redundant points are filtered using the E_optimum method, ensuring coverage is preserved while further reducing path length.

The main distinction among the approaches lies in the methods applied at the second and third stages. In particular, they differ in the type of geometric heuristics and TSP optimization methods.

To solve the above-described CETSP problem, seven different approaches are proposed. Each approach is based on a combination of several methods or algorithms.

2.2.1. The "Three-dot Line" approach

The first approach, referred to as the "Three-dot Line," involves a three-stage methodology.

Stage 1. An optimal route is constructed by solving the TSP over the target points located at the centers of the sensor coverage zones. Numerous algorithms exist for solving the TSP [14]. At this stage, any of these algorithms or their

combinations can be applied. Section 3 of this article presents comparative experimental results of various TSP algorithms applied at this stage.

Stage 2. Based on geometric calculations, for each sequence of three consecutive points in the route, the actual visiting points are determined using six geometric conditions [15, 16]. The method used at this stage is referred to as GEO-6. As a result, the trajectory is significantly shortened by constructing *E*-points that replace the original centers. The UAV is not required to pass through the center of the sensor's coverage zone – it is sufficient to pass through the interior or boundary of the zone.

Stage 3. In this stage, route optimization is performed based on reducing the number of *E*-points determined by the GEO-6 method. If the straight segment formed between the intermediate points $E(i)$ and $E(i+k)$ passes through the coverage area of the sensor corresponding to any intermediate point $E(j)$, where $i < j < k$, then point $E(j)$ is removed from the route list. As a result, the remaining points in the route list form a final trajectory constructed from the set of points known as E_optimum. The method used in this stage is called E_optimum. The full implementation of this approach is provided in Algorithm 1.

Algorithm 1. Three-dot Line approach for UAV trajectory optimization

Input: Sensor zone centers $P = \{P\_1, P\_2, ..., P\_n\}$, radii $R = \{r\_1, ..., r\_n\}$

Output: Effective UAV path *T*

Step 1. Solve TSP using Optimal Insertion

TSP_path ← Optimal Insertion TSP(*P*)

Step 2. Generate E-points from triplets in the TSP path

E_points ← []

for each triplet (*A*, *B*, *C*) in TSP_path:

  Determine which geometric rule applies:

  if angle $BCA \geq 90°$:

    E ← perpendicular from *B* to line *AC*

  else if angle $BAC \geq 90°$:

    E ← perpendicular from *A* to line *BC*

  else if angle $ABC \geq 90°$:

    E ← perpendicular from *C* to line *AB*

  else if only 2 circles intersect:

    E ← circle intersection point

  else if all 3 circles intersect:

    E ← center of common intersection

  else:

    E ← perpendicular from *B* to line *AC*

  Append *E* to E_points

Step 3. Filter redundant *E*-points

for each E_i with neighbors E_{i–1}, E_{i+1}:

  Z1 ← zones covered by (E_{ i–1}, E_i)

  Z2 ← zones covered by (E_i, E_{i+1})

  Z3 ← zones covered by (E_{ i–1}, E_{i+1})

  if Z3 covers all zones in Z1 ∪ Z2:

mark E_i as redundant
Filtered E ← unmarked E-points
unmarked E-points=E_optimum
Step 4. Compute final trajectory and metrics
T ← Path through E_optimum

Algorithm 1. A pseudo-code of an algorithm that implements the "Three dot Line" approach.

2.2.2. The "Overlap Three-Point" approach

The second proposed approach, termed the "Overlap Three-Point" approach, draws conceptual inspiration from the work presented in [17] and is designed to enhance trajectory efficiency by leveraging zone overlaps and geometric refinements. This algorithm follows a four-stage methodology. The key differences lie in the use of alternative methods in Stages 2-4.

Stage 1. In this stage, the UAV coverage model assumes that sensors are located within circular communication zones of known radii. When two or more such zones intersect, their union forms a common region in which the UAV can communicate with all involved sensors simultaneously. To exploit this redundancy, the algorithm identifies groups of overlapping circles and computes the largest inscribed circle within each group that is fully contained in the intersection area. The center of this inscribed circle becomes a new representative target point, replacing all the original centers in the group. This merging process reduces the number of required stops and thus shortens the total trajectory. All original nodes belonging to the merged region are removed from the list of targets, and only the unified center is retained.

Stage 2. With the updated and reduced set of target points obtained in Stage 1, a TSP is solved to generate a near-optimal visiting sequence. Any well-established TSP Algorithm or a combination thereof can be employed in this stage. Section 3 provides a comparative performance evaluation of various TSP solutions used for this stage.

Stage 3. For each sequence of three points in the route, geometric calculations are performed to determine refined visiting points. Two alternative methodologies are proposed: optimization based on bisector points (F-points), as described in [17], and determination of E-points using the GEO-6 method. In both methods, the resulting points ensure significant reduction in the total path length. As before, it is not required to pass through the center of the zone – it is sufficient for the UAV trajectory to intersect the interior or boundary of the circle.

Stage 4. Based on the E-points or F-points obtained in the previous stage, a final optimization is performed using the E_optimum method. If a line segment between two intermediate points passes through the coverage zone of another sensor, the corresponding point is removed. The result is a final, shortened trajectory based on the optimized set of E_optimum points. The pseudo-code of the algorithm that implements this approach is shown in Algorithm 2.

Algorithm 2. Algorithmic sequence for the second approach

Input: sensor_zones = $\{(x_1, y_1, r_1), (x_2, y_2, r_2), ..., (x_n, y_n, r_n)\}$ – set of sensor coverage zones

Output: M – optimal UAV trajectory passing through essential E-points

Step 1. Merge overlapping zones (three or more intersections)

For each triplet of sensor zones (C_i, C_j, C_k):

- If $C_i \cap C_j \cap C_k \neq \emptyset$ and the intersection area > ε:
  - Compute intersection polygon
  - Calculate centroid and equivalent radius
  - Create merged zone $C' = (x', y', r')$
  - Add C' to the final zone list
  - Mark C_i, C_j, C_k as merged

Step 1.2. Merge remaining overlapping pairs

For each unmerged pair (C_i, C_j):

- If $C_i \cap C_j \neq \emptyset$ and area > ε:
  - Compute intersection
  - Derive centroid and equivalent radius
  - Add new merged zone to final list
  - Mark C_i, C_j as merged

Step 1.3. Filter redundant zones

For each pair of overlapping merged zones (C_i, C_j):

- Keep only the zone with the larger radius
- Remove the smaller one

→ Result: filtered_zones – dominant, non-overlapping zones

Step 2. Build initial trajectory using greedy NN

Initialize route:

- route ← [(0, 0)]

While unvisited zones remain:

- For each zone:
  - Compute E-point (boundary point directed from current UAV location)
  - Calculate distance from current location
- Select nearest E-point
- Add E-point to route
- Update current location

Step 2.1. Optimize path with 2-OPT

Repeat until no further improvement:

- For all pairs of segments (i, j):
  - If reversing segment [i:j] reduces total length:
    - Swap (reverse) the segment

Step 3-4. Filter redundant E-points

Initialize:

- marked ← ∅

For each segment (E_i, E_j) in the path:

- Identify sensor zones covered by segment [E_i, E_j]
- For all E-points between E_i and E_j:

• If their associated zones are already covered:
• Mark them as redundant

→ Result: unmarked_E_points — essential trajectory waypoints

Step 5. Final trajectory construction

• Construct M = [(0, 0)] + unmarked_E_points + [(0, 0)]
• Compute total trajectory length
• Visualize:
• Final trajectory path
• Coverage circles for each zone
• Marked E-points along the route

Final output:

• M – optimized UAV path visiting all required sensor coverage areas with minimal length.

Algorithm 2. Pseudo-code of an algorithm that implements the "Overlap Three-Point" approach.

2.2.3. The Within_r approach

The Within_r approach is based on utilizing the coverage radii of sensor zones. The UAV trajectory is constructed with respect to the boundary of each zone's coverage area [18]. This approach consists of four stages, many of which are similar to those described in the "Overlap Three-Point" approach. Specifically, the first, third, and fourth stages are identical to the corresponding stages in the aforementioned approach.

In the second stage, the Within_r method is implemented. According to this method, for each sensor zone, the closest point on its boundary is calculated – that is, a point on the circumference defined by the coverage radius r corresponding to that zone.

If the current UAV position is denoted as $P_k = (x_k, y_k)$, then for each zone center $C = (x_i, y_i)$, the boundary point E_i is determined as follows:

Direction vector:

$$\theta_i = \arctan2(y_i - y_k,\ x_i - x_k);$$

Boundary point:

$$E_i = (x_i - r_i\cos\theta_i,\ y_i - r_i\sin\theta_i).$$

2.2.4. The Sampling approach

The Sampling approach is based on selecting an arbitrary point located on the boundary of the sensor's coverage zone [19-21]. The trajectory points are selected from a predefined set of sampling points located along the boundary of each sensor zone. From this set, eight points are chosen per zone (a larger number can be used, but this significantly increases computation time). The UAV trajectory is then constructed by selecting one of these candidate points. This approach follows a four-stage process similar to the ones in previously described approaches. The first and fourth stages employ the same methods as in the earlier approaches.

Stage 2. For each remaining or merged sensor zone, eight equidistant points are selected along the circumference of the zone's coverage circle. These sampling

points serve as candidate UAV entry points for the trajectory construction. As a result, the total number of candidate points E is eight times the number of sensor zones,

$$E_{ij} = \left(x_i + r_i\cos\theta_j,\ y_i + r_i\sin\theta_i\right),\ \theta_j = \frac{2\pi j}{8},\ j = 0, \dots, 7.$$

Here, E_{ij} denotes the *j*-th sampling point within the *i*-th sensor zone.

Stage 3. The TSP is solved considering eight discrete points sampled along the boundary of each sensor's coverage zone. The trajectory is constructed based on the selected point from this set, and the optimal visiting points *E* are determined according to the mathematical logic of the algorithm. The authors propose using the Nearest Neighbor Algorithm (NNA) at this stage. Specifically, when applying the NNA, the corresponding point *E* is selected using the following expression:

$$E^* = \arg\min_{E \in U} \|E - P_k\|.$$

Here: *U* is the set of eight candidate points in the zones that have not yet been visited; P_k is the last point in the currently constructed trajectory.

2.2.5. The Convex Hull approach

The Convex Hull approach provides a geometric strategy for selecting UAV trajectory points within the coverage zones by utilizing the convex envelope of the initial point set. This approach is particularly efficient in sparse or semi-structured sensor deployments, where many sensor zones lie near the periphery of the monitored region [22-23]. The key idea is to begin with a convex hull path through a subset of the outermost zones and then iteratively insert remaining points using cost-based heuristics.

As with earlier approaches, this approach follows a four-stage process, with the first and fourth stages identical to those of the "Overlap Three-Point" approach.

Stage 2. Selection of *E*-points – that is, formation of the initial set of trajectory points. For each sensor zone, $N = 8$ discrete points are selected along the circumference (uniformly spaced by angle). These points are computed along the radius as follows:

$$E_i^j = \left(x_i + r_i\cos\left(\frac{2\pi j}{N}\right),\quad y_i + r_i\sin\left(\frac{2\pi j}{N}\right)\right),\quad j = 0, \dots, N-1.$$

Here, E_i^j denotes the *j*-th point in the *i*-th sensor zone. From all discretely sampled points, one point is randomly selected to represent each zone.

$$P_{\text{init}} = \{P_1, P_2, \dots, P_n\},\ P_i \in \{E_i^0, E_i^1, \dots, E_i^{N-1}\}.$$

An initial path is then constructed using the following geometric principle:

$$H = \text{ConvexHull}(P_{\text{init}}).$$

Here: P_{init} refers to one point taken from the boundary of each zone, i.e., one discretized point selected from each zone; *H* represents the convex hull formed by these points, that is, the outermost boundary points.

As a result, the initial trajectory is defined as

$$T_0 = \{P_0, H_1, H_2, \dots, H_k, P_0\}.$$

Stage 3. At this stage, the TSP is solved based on the identified *H* points. One or several TSP algorithms can be applied. The authors propose using the

Optimal Insertion Algorithm (OIA) followed by the 2-OPT Algorithm as the optimal solution. For each zone, the following insertion cost is calculated:

$$\Delta L = \|P_i - E_j\| + \|E_j - P_{i+1}\| - \|P_i - P_{i+1}\|.$$

The point E_j is inserted between P_i and P_{i+1} at the position that results in the smallest ΔL. The constructed route is then further optimized using the 2-OPT Algorithm. At this stage, two segments in the route are swapped to reduce the overall trajectory length. If the following condition is met:

$$\|P_i - P_j\| + \|P_{i+1} - P_{j+1}\| < \|P_i - P_{i+1}\| + \|P_j - P_{j+1}\|,$$

then: $[P_{i+1}, ..., P_j] \rightarrow$ reverse.

2.2.6. The Directional Nearest Neighbor (DNN) approach

This approach is inspired by the DNN approach presented in the paper by [24]. In other words, this approach can be viewed as a modified version of the original method. It is similar in structure to the previously described Within_r approach, but differs in its algorithmic implementation. The objective of the DNN Algorithm is to construct a faster and more efficient UAV route by allowing entry into a sensor zone through the nearest boundary point rather than through its center. This helps reduce the total mission time and improves data collection efficiency. The structure of the proposed approach includes steps analogous to the first and fourth stages described in earlier approaches and uses the same methods applied in those stages.

Stage 2. Selection of *E*-points, i.e., the initial set of route points. Let the initial UAV position be $p_0 = (0, 0)$ and the set of all sensor zones be $\{C_1, C_2, ..., C_n\}$, each with a coverage radius $\{r_1, r_2, ..., r_n\}$. At the first step, the nearest zone center C_i is determined as follows:

$$j = \underset{i \in U}{\operatorname{argmin}} \|C_i - P_k\|.$$

Finding a point on the edge of this zone, i.e., finding a route point,

$$E_j = C_j - \frac{r_j \times (C_j - P_k)}{\|C_i - P_k\|}.$$

The UAV moves to the selected point: $P_{k+1} = E_j$. Zone *j* is added to the list of visited zones. This process repeats until all zones have been covered.

Stage 3. A TSP is then solved over the set of determined E_j-points. One or several TSP algorithms can be applied. The authors recommend first applying the NNA, followed by the 2-OPT Algorithm to further optimize the trajectory.

2.2.7. The Directional Directed to the next Nearest Node (DDNN) approach

This approach is based on the DDNN strategy described in [24]. The DDNN approach is similar to the previously described DNN approach but introduces key algorithmic differences. It represents an improved version of the Nearest Neighbor method in which the UAV's approach point to each sensor zone is directed beyond the center of the circle, toward the direction of the next nearest zone. The core idea of this approach is to modify the classical TSP-NN Algorithm so that, at the beginning of its movement: The UAV moves directly to the first selected node; upon reaching the edge of the node's communication radius, it targets a boundary

point on the coverage circle, which is treated as a turning point; then it proceeds toward the next closest node, and so on.

This process continues until all nodes have been visited. This approach is similar to the first and fourth stages in the second approach. It uses the methods in these stages.

Stage 2. First, the nearest zone center is selected:

$$P = \arg\min_{(x_i,y_i)\in U} \|(x_i, y_i) - p_0\|.$$

Then, the entry point Q on the outer boundary of zone P is calculated as

$$Q = P - \frac{r_p \times P - B}{\|P - B\|}.$$

This point lies on the edge of the coverage circle. The UAV is assumed to pass through point Q, and zone P is marked as visited: $U = U \backslash \{P\}$. Updated current position: $B = P$.

The next closest zone A in the queue is selected using:

$$A = \arg\min_{(x_i,y_i)\in U} \|(x_i, y_i) - B\|.$$

The new turning point C is computed as

$$C = A - \frac{r_a \times A - Q}{\|A - Q\|}.$$

The UAV is then assumed to pass through point C, and zone A is marked as visited: $U = U \backslash \{A\}$. Updated current positions: $Q = C$, $B = A$.

The resulting set of route points is $T_0 = \{P_0, Q_1, C_1, C_2, \ldots, C_k, P_0\}$.

Stage 3. Based on the set of generated route points T_0, a TSP is formulated. For solving this problem, one may use any known TSP algorithm or a hybrid method, i.e., a combination of two algorithms. The authors propose applying the NNA, followed by the 2-OPT Algorithm for further route optimization.

For consistency and clarity in the evaluation of the proposed approaches, the concept of efficiency in UAV trajectory is defined as a composite measure comprising four indicators: the total path length traveled by the UAV [39], the computational cost expressed in CPU time [40], the number of maneuver points required along the trajectory [41], and the relative deviation of the planned trajectory from the sensor centers [10, 42]. Taken together, these indicators provide a comprehensive and quantifiable framework that corresponds to the formulated research tasks and establishes a transparent basis for the comparative assessment of the proposed methods.

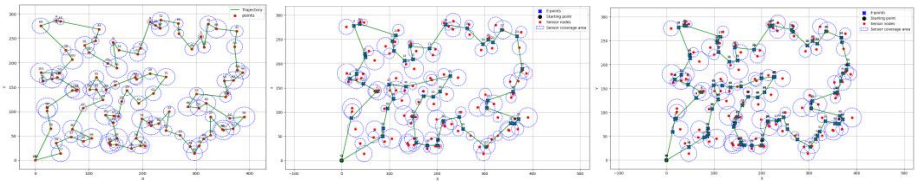
3. Numerical results

This section presents the numerical results obtained by applying the developed algorithms, based on the proposed approaches, to the CETSP. Experiments were conducted using a common set of 100 sensor zones for each of the proposed algorithms. Table 1 provides the coordinates and coverage radii of the 100 sensor zones used across all experiments.

Table 1. Parameters of the 100 sensor zones deployed within a 400 × 300 unit monitoring area

No	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
x	359	78	39	309	205	212	214	145	161	155	363	14	47	179	112
y	137	145	287	254	29	77	284	242	121	90	86	163	285	24	245
Radius	5	5	5	5	5	5	5	6	6	6	7	7	7	7	7
No	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
x	89	314	149	224	135	132	145	268	270	267	327	57	171	327	244
y	107	44	154	277	199	145	77	260	51	281	89	226	143	51	61
Radius	7	8	8	8	8	8	8	9	9	9	10	10	10	10	11
No	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**
x	250	355	39	100	101	67	315	214	230	366	41	189	119	90	387
y	100	130	35	221	145	48	232	178	75	259	170	166	269	39	180
Radius	11	11	11	11	12	12	12	12	12	12	12	12	12	12	13
No	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**
x	146	289	292	369	151	196	301	353	114	188	69	186	45	349	220
y	91	28	228	65	32	230	136	163	170	45	143	170	179	63	137
Radius	13	13	13	13	13	13	13	14	14	14	14	14	14	15	15
No	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**61**	**62**
x	191	372	193	297	306	233	377	10	323	22	234	305	156	191	372
y	218	233	117	14	30	271	185	276	280	101	288	107	226	218	233
Radius	15	15	15	15	16	16	16	16	16	16	16	16	16	15	15
No	**74**	**75**	**76**	**77**	**78**	**79**	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**
x	244	29	47	196	126	69	92	140	370	138	319	285	76	338	216
y	171	65	14	30	124	64	236	32	164	36	117	110	144	270	71
Radius	17	17	17	17	18	18	18	18	18	18	18	19	19	19	19
No	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**					
x	114	131	105	390	152	152	22	351	69	11					
y	145	93	45	89	189	45	108	62	207	180					
Radius	19	20	20	20	20	20	20	20	20	20					

Fig. 2 provides a visual representation of the sensor coverage zones distributed over a 400×300 unit area, as defined by the data presented in Table 1. Fig. 2a specifically illustrates the solution to the TSP constructed using the OIA, where the UAV trajectory is computed based on the coordinates of the centers of the sensor zones. This result corresponds to stage 1 of the “three-dot line” approach, where the initial route is generated by treating the centers of the zones as target points. In the first stage of this approach, an initial UAV trajectory is constructed based on the sensor zone centers, resulting in a total path length of 2758 units. This route effectively intersects all designated sensor coverage zones and can therefore be considered suitable for data collection tasks. In the second stage, the trajectory is refined using *E*-points determined by the GEO-6 method, which incorporates geometric rules to reduce unnecessary detours. This refinement decreases the total trajectory length to 1961 units, with 98 E-points generated to ensure coverage.



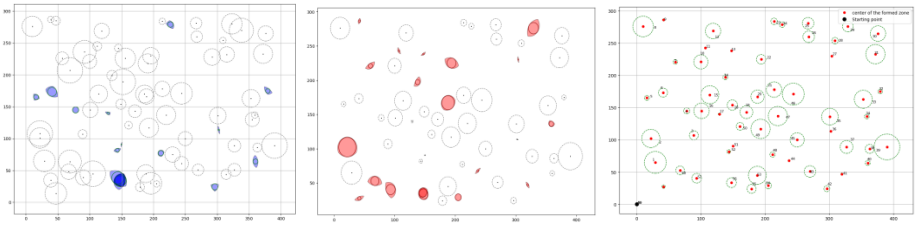
(a) (b) (c)

Fig. 2. Trajectories built for sensor zones based on the “three-dot line” approach: Trajectory generated using the OIA based on the centers of the initial sensor zones (a); Trajectory optimization using the GEO-6 method applied to the route points formed from sensor zone centers (b); Efficient trajectory covering the sensor zones, constructed using the three-dot line approach (c)

In the third stage, redundant *E*-points are eliminated based on coverage overlap and segment intersection logic. The resulting optimized trajectory is illustrated in Fig. 2c, with a final path length of 1934 units. This final configuration successfully covers all sensor zones using only 52 maneuver points, defined as the critical turning or shaping points of the trajectory.

- Experiment for the Second Approach

The “Overlap Three-Point” approach to solving the CETSP consists of four stages, with Stage 1 involving a series of operations that generate merged and isolated sensor zones. The general structure of Stage 1 of this approach is illustrated in Fig. 3.



(a) (b) (c)

Fig. 3. The results of the first stage of processing, taking into account the partial overlap of the coverage areas: First stage of the overlap operation – sample visualization of regions formed by the intersection of three or more circles (a); Visualization of shapes obtained from the intersection of two neighboring circles (b); Final distribution of sensor zones after completion of the first stage (c)

The first stage of the “Overlap Three-Point” approach consists of the overlap operation, which includes several sequential procedures. Initially, the common intersection region of three or more circles is identified. In Fig. 3a, the resulting areas are shown in blue. For each such region, the maximum inscribed circle is computed along with its center. Since this step focuses on intersections involving three or more zones, the corresponding original sensor zones are removed from the list, and the newly formed zones (circles) are added to it. Next, the second overlap step is performed for pairs of intersecting circles. As shown in Fig. 3b, a common intersection area is identified for each pair, and within it, the largest inscribed circle is generated, and its center is calculated. The original pair of circles is removed from the list, and the new circle is added. As a result, an updated set of sensor zones is formed over the 400×300 unit area, illustrated in Fig. 3c. After applying the

overlap operations to the initial 100 sensor zones, accounting for their mutual intersections, a total of 58 merged sensor zones are obtained. Based on these merged zones, the second stage of the proposed approach is carried out: a TSP is solved over the centers of the 58 circles using the 2-OPT Algorithm. The result is shown in Fig. 4a. The total length of the trajectory formed in the second stage is 2379 units. While this trajectory can serve as a feasible solution to the problem, further optimization is possible. In the third stage, using the route obtained in the second stage, sequences of three consecutive points (*A–B–C*) are analyzed to determine bisector points. A refined trajectory is then constructed based on these bisector points (Figs 4b-4c). The total length of the trajectory constructed at this stage by the most effective version of the algorithm is 2017 units, with 58 trajectory points. To further improve the efficiency of the trajectory obtained in the third stage of the "Overlap Three-Point" approach, a fourth stage is carried out. At this stage, the E_optimum method is applied, which reduces the number of trajectory points (from 58) while ensuring that the path still intersects all coverage zones. The result is shown in Fig. 4d.

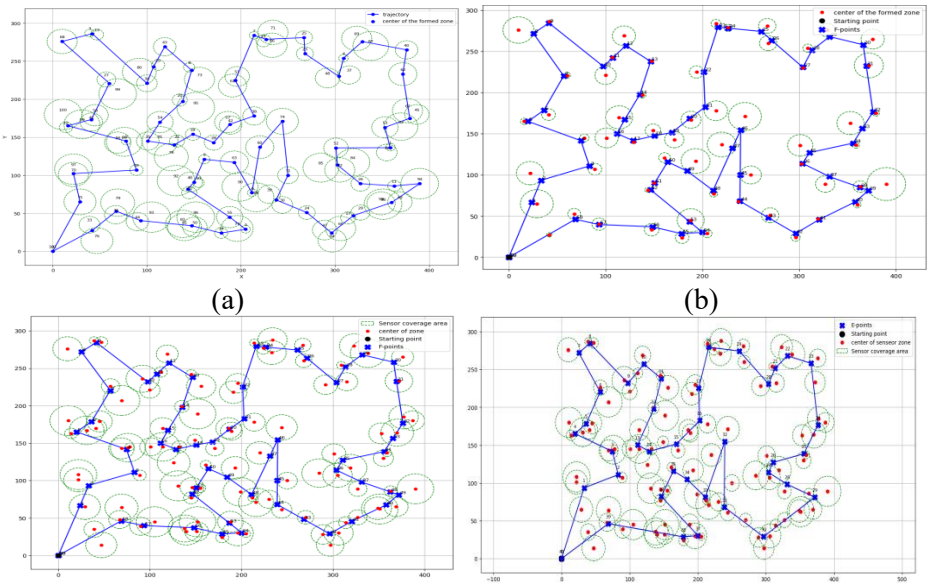


Fig. 4. Visualization of trajectory construction at different stages of the "Overlap Three-Point" approach: Initial trajectory through sensor zone centers using the 2-OPT Algorithm (Stage 1) (a); Refined trajectory over isolated zones formed during Stage 1 (b); Trajectory adjusted based on the original sensor zones (Stage 3) (c); Final optimized trajectory covering all sensor zones using the complete "Overlap Three-Point" approach (d)

- Experiment for the Third Approach

The experiment for the "Within_r" approach was conducted according to the four stages described in Section 2. The construction of the complete trajectory involved the following methods: overlap, within_r, nearest neighbor, 2-OPT, and E_optimum.

- Experiment for the Fourth Approach

The experiment for the Sampling approach was conducted using three stages (excluding the first of the four stages described in Approach 2). The overall trajectory was constructed using the following methods: Sampling, Nearest Neighbor (NN), 2-OPT, and E_optimum.

- Experiment for the Fifth Approach

The experiment for the Convex Hull approach was also conducted using three stages (excluding the first of the four mentioned previously). The overall trajectory was constructed using the following methods: Convex Hull, Optimal Insertion, 2-OPT, and E_optimum.

- Experiment for the Sixth Approach

The experiment for the DNN approach was conducted using all four stages. The overall trajectory was constructed using the following methods: Overlap, DNN, NN, 2-OPT, and E_optimum.

- Experiment for the Seventh Approach

The experiment for the **DDNN** approach was carried out using the complete **four-stage process**. The overall trajectory was constructed using the following sequence of methods: **Overlap**, **DDNN**, **Nearest Neighbor**, **2-OPT**, and **E_optimum**.

- Results analysis

Based on the seven proposed approaches, **hybrid algorithms** were designed and implemented as part of this study. Each algorithm was developed in Python and executed in a cloud-based Google Colab environment using an Intel Xeon processor with 12.67 GB of RAM (via Google Compute Engine).

All experiments were conducted using the common set of 100 sensor zones provided in Table 1. The experimental results were evaluated across four key parameters: trajectory length, algorithm computation time, total number of maneuver points in the generated trajectory, and relative proximity to each sensor zone. The results for these four parameters are summarized in Table 2.

Table 2 presents a comparative analysis of the efficiency of various hybrid algorithms developed using the proposed methodological approaches. In total, **92 hybrid algorithm variants** were implemented – each representing a **unique combination** of routing and optimization strategies.

The notations and abbreviations in the table represent key components of the hybrid algorithms. Optimal Insertion (OI) is a heuristic path-building method from [25], while NN uses proximity-based logic [26]. TSI (Two-Step Insertion) extends basic insertion by considering two-point integration [27]. The 2-OPT Algorithm [28] refines trajectories by removing intersecting segments. The Overlap method, applied in the first stage, merges intersecting sensor zones to simplify planning. In the third stage, the Bisser method introduces bisector points (from nodes *A*, *B*, and *C*) to improve route geometry. GEO-6 generates *E*-points that smooth sharp turns. In the final stage, E_optimum removes redundant maneuver points while maintaining complete zone coverage, reducing path complexity without performance loss.

Table 2. Analytical results of the performance of 92 hybrid algorithms developed based on seven proposed approaches

C	Approaches (algorithms)	*L*	*N*	*T*	*R/d*
1.1	Three-dot line (OI+GEO-6+E_optimum)	1934	53	7.2	0.69
1.2	Three-dot line (2-OPT+GEO-6+E_optimum)	2077	51	16	0.64
1.3	Three-dot line (NN+GEO-6+E_optimum)	2490	53	9.2	0.6
1.4	Three-dot line (TSI+GEO-6+E_optimum)	2343	62	8.9	0.69
1.5	Three-dot line (TSI+2-OPT+GEO-6+E_optimum)	2087	56	11.7	0.67
1.6	Three-dot line (OI+2-OPT+GEO-6+E_optimum)	2219	53	9.8	0.64
2.1	Overlab three point (overlab+OI+Bisser+E_optimum)	2183	47	22	0.64
2.2	Overlab three point (overlab+2-OPT+Bisser+E_optimum)	2000	40	72	0.58
2.3	Overlab three point (overlab+NN+Bisser+E_optimum)	2633	46	22	0.6
2.4	Overlab three point (overlab+TSI+Bisser+E_optimum)	2047	43	22	0.62
2.5	Overlab three point (overlab+TSI+2-OPT+Bisser+E_optimum)	2443	41	19	0.55
2.6	Overlab three point (overlab+OI+2-OPT+Bisser+E_optimum)	2143	43	21	0.62
2.7	Overlab three point (overlab+OI+Geo-6+E_optimum)	2210	50	24	0.63
2.8	Overlab three point (overlab+2-OPT+Geo-6+E_optimum)	2017	44	81	0.59
2.9	Overlab three point (overlab+NN+Geo-6+E_optimum)	2644	46	25	0.61
2.10	Overlab three point (overlab+TSI+Geo-6+E_optimum)	2070	47	20	0.63
2.11	Overlab three point (overlab+TSI+2-OPT+Geo-6+E_optimum)	2467	40	23.2	0.57
2.12	Overlab three point (overlab+OI+2-OPT+Geo-6+E_optimum)	2158	43	25.7	0.63
3.1.1	Within_r (Within_r +NN+E_optimum)	2613	53	6	0.56
3.1.2	Within_r (Within_r +NN+2-OPT+E_optimum)	2574	53	13	0.57
3.1.3	Within_r (overlab+Within_r +NN+E_optimum)	2594	53	10	0.58
3.1.4	Within_r (overlab+Within_r +NN+2-OPT+E_optimum)	2282	51	14	0.58
3.2.1	Within_r (Within_r +OI+E_optimum)	2844	48	4	0.56
3.2.2	Within_r (Within_r +OI+2-OPT+E_optimum)	2499	46	42	0.6
3.2.3	Within_r (overlab+Within_r +OI+E_optimum)	2681	49	12	0.5
3.2.4	Within_r (overlab+Within_r +OI+2-OPT+E_optimum)	2515	49	15.3	0.51
3.3.1	Within_r (Within_r +TSI+E_optimum)	2650	60	8.86	0.52
3.3.2	Within_r (Within_r +TSI+2-OPT+E_optimum)	2492	52	22.6	0.52
3.3.3	Within_r (overlab+Within_r +TSI+E_optimum)	2369	50	9.83	0.53
3.3.4	Within_r (overlab+Within_r +TSI+2-OPT+E_optimum)	2297	49	11.4	0.53
4.1.1	Sampling (sampling+NN+E_optimum)	2462	51	3	0.55
4.1.2	Sampling (sampling+NN+2-OPT+E_optimum)	2323	48	18	0.52
4.1.3	Sampling (overlab+sampling+NN+E_optimum)	2507	54	9	055
4.1.4	Sampling (overlab+sampling+NN+2-OPT+E_optimum)	2368	49	13	0.55
4.2.1	Sampling (sampling+OI+E_optimum)	2590	51	32	0.56
4.2.2	Sampling (sampling+OI+2-OPT+E_optimum)	2553	50	40	0.57
4.2.3	Sampling (overlab+sampling+OI+E_optimum)	2477	48	19	0.55
4.2.4	Sampling (overlab+sampling+OI+2-OPT+E_optimum)	2373	49	19	0.56
4.3.1	Sampling (sampling+TSI+E_optimum)	2980	51	2.3	0.5
4.3.2	Sampling (sampling+TSI+2-OPT+E_optimum)	2645	52	2.3	0.5
4.3.3	Sampling (overlab+sampling+TSI+E_optimum)	2633	46	7.2	0.49
4.3.4	Sampling (overlab+sampling+TSI+2-OPT+E_optimum)	2524	47	6.82	0.48
4.4.1	Sampling (sampling+2-OPT+E_optimum)	2480	53	30	0.58
4.4.2	Sampling (overlab+sampling+2-OPT+E_optimum)	2392	50	17	0.51
5.1.1	Convex Hull (Convex Hull +OI+E_optimum)	2351	58	7.5	0.68

Table 2 (c o n t i n u e d)

5.1.2	Convex Hull (Convex Hull +OI+2-OPT+E_optimum)	2232	56	14.6	0.7
5.1.3	Convex Hull (overlab+ Convex Hull +OI+E_optimum)	2360	49	8	0.53
5.1.4	Convex Hull (overlab+ Convex Hull +OI+2-OPT+E_optimum)	2331	49	11.2	0.54
5.2.1.	Convex Hull (Convex Hull +NN+E_optimum)	4190	57	7.1	0.51
5.2.2	Convex Hull (Convex Hull +NN+2-OPT+E_optimum)	3084	58	23	0.5
5.2.3	Convex Hull (overlab+ Convex Hull +NN+E_optimum)	3645	52	12	0.48
5.2.4	Convex Hull (overlab+ Convex Hull +NN+2-OPT+E_optimum)	2683	49	13.4	0.47
5.3.1	Convex Hull (Convex Hull +TSI+E_optimum)	2784	64	3.4	0.56
5.3.2	Convex Hull (Convex Hull +TSI+2-OPT+E_optimum)	2627	57	4.3	0.55
5.3.3	Convex Hull (overlab+ Convex Hull +TSI+E_optimum)	2579	51	8.6	0.48
5.3.4	Convex Hull (overlab+ Convex Hull +TSI+2-OPT+E_optimum)	2473	50	14.1	0.5
5.4.1	Convex Hull (Convex Hull +2-OPT+E_optimum)	3648	62	6.2	0.49
5.4.2	Convex Hull (overlab+ Convex Hull +2-OPT+E_optimum)	3241	51	12.3	0.6
6.1.1	DNN (DNN +OI+E_optimum)	2645	53	15.6	0.5
6.1.2	DNN (DNN+OI+2-OPT+E_optimum)	2452	50	23	0.53
6.1.3	DNN (overlab+ DNN +OI+E_optimum)	2526	49	10.4	0.54
6.1.4	DNN (overlab+DNN+OI+2-OPT+E_optimum)	2367	49	13	0.54
6.2.1	DNN (DNN +TSI+E_optimum)	2547	49	5.3	0.52
6.2.2	DNN (DNN+TSI+2-OPT+E_optimum)	2299	46	6.1	0.58
6.2.3	DNN (overlab+ DNN +TSI+E_optimum)	2205	52	9.7	0.56
6.2.4	DNN (overlab+DNN+TSI+2-OPT+E_optimum)	2185	51	11.8	0.57
6.3.1	DNN (DNN +NN+E_optimum)	2656	52	6.4	0.54
6.3.2	DNN (DNN+NN+2-OPT+E_optimum)	2215	50	6.6	0.57
6.3.3	DNN (overlab+ DNN +NN+E_optimum)	2346	50	10.8	0.57
6.3.4	DNN (overlab+DNN+NN+2-OPT+E_optimum)	2127	47	15.4	0.57
6.4.1.	DNN (DNN +E_optimum)	2665	53	5.6	0.55
6.4.2	DNN (DNN+2-OPT+E_optimum)	2558	53	10	0.57
6.4.3	DNN (overlab+ DNN+E_optimum)	2460	52	10.7	0.58
6.4.4	DNN (overlab+DNN+2-OPT+E_optimum)	2346	50	11.6	0.58
7.1.1	DDNN (NN+DDNN+E_optimum)	2669	49	5.4	0.55
7.1.2	DDNN (DDNN+NN+E_optimum)	2799	51	10	0.49
7.1.3	DDNN (DDNN +NN+2-OPT+E_optimum)	2412	51	30	0.51
7.1.4	DDNN (overlab+NN+DDNN+E_optimum)	2743	45	13	0.54
7.1.5	DDNN (overlab+DDNN+NN+E_optimum)	2420	44	16	0.56
7.1.6	DDNN (overlab+DDNN+NN++2-OPT+E_optimum)	2256	45	17	0.59
7.2.1	DDNN (DDNN +TSI+E_optimum)	2752	50	7.2	0.49
7.2.2	DDNN (DDNN+TSI+2-OPT+E_optimum)	2620	51	15	0.46
7.2.3	DDNN (overlab+ DDNN +TSI+E_optimum)	2740	47	14.9	0.54
7.2.4	DDNN (overlab+DDNN+TSI+2-OPT+E_optimum)	2460	47	13.5	0.54
7.3.1	DDNN (DDNN +OI+E_optimum)	2672	52	19.3	0.47
7.3.2	DDNN (DDNN+OI+2-OPT+E_optimum)	2552	54	16.3	0.49
7.3.3	DDNN (overlab+ DDNN +OI+E_optimum)	2591	47	12	0.57
7.3.4	DDNN (overlab+DDNN+OI+2-OPT+E_optimum)	2523	45	12.6	0.56
7.4.1	DDNN (DDNN +E_optimum)	2681	50	11.3	0.54
7.4.2	DDNN (DNN+2-OPT+E_optimum)	2503	50	2.3	0.53
7.4.3	DDNN (overlab+ DDNN+E_optimum)	2743	45	7.45	0.54
7.4.4	DDNN (overlab+DDNN+2-OPT+E_optimum)	2582	48	7.25	0.56

The performance of the hybrid algorithms was evaluated using four key metrics: total trajectory Length (L), Number of maneuver points (N), computation

Time (T), and Relative deviation (R/d). Length L measures the overall UAV travel distance; N indicates the number of hotspots forming the route; T reflects algorithmic processing time; and R/d represents the normalized average distance from the UAV path to the centers of the sensor zones, directly impacting data accuracy.

Depending on mission constraints, the four metrics are prioritized as follows.

- For energy-constrained missions, the priority is L, followed by N, T, and finally R/d.
- For real-time or on-board replanning tasks, T is the primary criterion, followed by L, N, and R/d.
- Under strong kinematic constraints, N (or trajectory smoothness) is prioritized first, followed by L, T, and R/d.
- For coverage-quality-driven missions, R/d is the leading criterion, followed by L, N, and T.

For multi-objective settings, a normalized weighted score is additionally reported:

$$S = \omega_1 L' + \omega_2 T' + \omega_3 N' + \omega_4 \left(\frac{R}{d}\right)',$$

where L', T', N', $(R/d)'$ denote the normalized values of the metrics, and the weights ω_1, ω_2, ω_3, ω_4 are weighting coefficients that reflect the relative importance of the metrics under specific mission scenarios.

Communication quality, energy consumption, and real-time trajectory adjustments are beyond the scope of this study and left for future work. The results show that algorithm performance strongly depends on the spatial distribution and coverage radius of sensor zones. Besides coverage, factors such as node density, inter-node distances, and network topology also affect outcomes. Therefore, a preliminary assessment of sensor placement is recommended. In dense deployments, overlap-based approaches tend to perform better.

4. Conclusion

Based on experimental analysis, the efficiency of 92 hybrid algorithms – developed from seven convergence models – was evaluated according to four key parameters: total trajectory length, computation time, number of maneuver points, and relative deviation from sensor zones. The results demonstrated that trajectory performance depends not only on the chosen algorithm, but also on the placement, spatial density, and coverage radius of the sensor nodes. One of the key aspects of the analysis was the relative deviation parameter, reflecting how close the trajectory passes to the centers of the sensor zones – thus directly influencing data acquisition quality.

To further demonstrate the practical relevance of the proposed methods, a case study was conducted based on the dataset of 100 sensor zones deployed in a 400 × 300 unit area (Table 1 and Fig. 2). The configuration models a precision agriculture scenario with soil moisture and temperature sensors having coverage radii between 5 and 20 units. Using a classical TSP-based routing approach, the

UAV trajectory length was about 2600 units. With the proposed hybrid algorithms, the total route length was reduced by 15-20% while ensuring complete coverage of all sensor zones. This reduction leads to shorter flight times and lower energy consumption, which is particularly important for UAVs with limited battery capacity. Based on the above results, the "Three-dot Line" can be recognized as the most effective approach.

Although the proposed approaches demonstrate efficiency in UAV trajectories for large-scale data collection, certain practical constraints, such as UAV autonomy margin, communication range, and environmental factors, were not explicitly addressed in this study. These factors represent important practical considerations and will be investigated in future research to extend the applicability of the proposed approaches to real-world scenarios.

***Acknowledgment*:** The authors gratefully acknowledge the Urgench State University for its invaluable support and resources throughout this research.

References

1. Abramov, M. M. New and Promising Directions for the Use of Unmanned Aerial Vehicles. Izvestiya of Tula State University. – Technical Sciences, 2022, No 3, pp. 227-232.
2. Popescu, D., et al. A Survey of Collaborative UAV-WSN Systems for Efficient Monitoring. – Sensors, Vol. **19**, 2019, No 21, 4690.
3. Hedges, D. A., J. P. Coon, G. Chen. A Continuum Model for Route Optimization in Large-Scale Inhomogeneous Multi-Hop Wireless Networks. – IEEE Transactions on Communications, Vol. **68**, 2020, No 2, pp. 1058-1070.
4. Jawhar, I., N. Mohamed, J. Al-Jaroodi, S. Zhang. A Framework for Using Unmanned Aerial Vehicles for Data Collection in Linear Wireless Sensor Networks. – J. Intell. Robot. Syst., Vol. **74**, 2014, pp. 437-453.
5. Dumitrescu, A., J. S. B. Mitchell. Approximation Algorithms for TSP with Neighborhoods in the Plane. – Journal of Algorithms, Vol. **48**, 2003, No 1, pp. 135-159.
6. Alatartsev, S., M. Augustine, F. Ortmeier. Constricting Insertion Heuristic for Traveling Salesman Problem with Neighborhoods. – In: Proc. of International Conference on Automated Planning and Scheduling, Vol. **23**, 2013, pp. 2-10.
7. Gulczynski, D. J., J. W. Heath, C. C. Price. The Close Enough Traveling Salesman Problem: A Discussion of Several Heuristics. – Springer US, 2006, pp. 271-283.
8. Abramov, M. M. New and Promising Directions for the Use of Unmanned Aerial Vehicles, Izvestiya of Tula State University. – Technical Sciences, 2022, No 3, pp. 227-232.
9. Carrabs, F., et al. Improved Upper and Lower Bounds for the Close Enough Traveling Salesman Problem. – In: Proc. of 12th International Conference, Green, Pervasive, and Cloud Computing (GPC'17), Cetara, Italy, 11-14 May 2017. Springer International Publishing, 2017, pp. 165-177.
10. Qian, Q., Y. Wang, D. Boyle. On Solving Close Enough Orienteering Problems with Overlapped Neighborhoods. – European Journal of Operational Research, Vol. **318**, 2024, No 2, pp. 369-387.
11. Carrabs, F., et al. A Novel Discretization Scheme for the Close Enough Traveling Salesman Problem. – Computers & Operations Research, Vol. **78**, 2017. pp. 163-171.
12. Carrabs, F., et al. An Adaptive Heuristic Approach to Compute Upper and Lower Bounds for the Close-Enough Traveling Salesman Problem. – INFORMS Journal on Computing, Vol. **32**, 2020, No 4, pp. 1030-1048.
13. Sinha Roy, D., et al. Estimating the Tour Length for the Close Enough Traveling Salesman Problem. – Algorithms, Vol. **14**, 2021, No 4, p. 123.

14. A i t S a a d i, A., et al. UAV Path Planning Using Optimization Approaches: A Survey. – Archives of Computational Methods in Engineering, Vol. **29**, 2022, No 6, pp. 4233-4284.
15. R o d i o n o v, A. S., T. A. M a t k u r b a n o v. UAV Flight Trajectory Planning for Large-Area Monitoring. – Informatics and Automation, Vol. **24**, 2025, No 3, pp. 791-827.
16. M a r k o v, A. V., V. I. S i m a n k o v. Methodology for Calculating UAV Flight Trajectories for Terrain Observation. – Reports of BSUIR, Vol. **4**, 2019, No 122, pp. 57-63.
17. C a r i o u, C., et al. Evolutionary Algorithm with Geometrical Heuristics for Solving the Close Enough Traveling Salesman Problem: Application to the Trajectory Planning of an Unmanned Aerial Vehicle. – Algorithms, Vol. **16**, 2023, No 1, p. 44.
18. A l e m a y e h u, T. S., J. H. K i m. Efficient Nearest Neighbor Heuristic TSP Algorithms for Reducing Data Acquisition Latency of UAV Relay WSN. – Wireless Personal Communications, Vol. **95**, 2017, No 3, pp. 3271-3285.
19. I s a a c s, J. T., J. P. H e s p a n h a. Dubins Traveling Salesman Problem with Neighborhoods: A Graph-Based Approach. – Algorithms, Vol. **6**, 2013, No 1, pp. 84-99.
20. J a n s o n, L., B. I c h t e r, M. P a v o n e. Deterministic Sampling-Based Motion Planning: Optimality, Complexity, and Performance. – The International Journal of Robotics Research, Vol. **37**, 2018, No 1, pp. 46-61.
21. K a r a m a n, S., E. F r a z z o l i. Sampling-Based Algorithms for Optimal Motion Planning. – The International Journal of Robotics Research, Vol. **30**, 2011, No 7, pp. 846-894.
22. A l a t a r t s e v, S., M. A u g u s t i n e, F. O r t m e i e r. Constricting Insertion Heuristic for Traveling Salesman Problem with Neighborhoods. – In: Proc. of International Conference on Automated Planning and Scheduling, Vol. **23**, 2013, pp. 2-10.
23. G o u t h a m, M., et al. A Convex Hull Cheapest Insertion Heuristic for the Non-Euclidean TSP. – arXiv Preprint arXiv:2302.06582, 2023.
24. A l e m a y e h u, T. S., J. H. K i m. Efficient Nearest Neighbor Heuristic TSP Algorithms for Reducing Data Acquisition Latency of UAV Relay WSN. – Wireless Personal Communications, Vol. **95**, 2017, No 3, pp. 3271-3285.
25. J o h n s o n, D. S., L. A. M c G e o c h. The Traveling Salesman Problem: A Case Study in Local Optimization. – In: Local Search in Combinatorial Optimization, 1997.
26. H o u g a r d y, S., M. W i l d e. On the Nearest Neighbor Rule for the Metric Traveling Salesman Problem. – In: Discrete Applied Mathematics, 2015.
27. R o d i o n o v, A. S., T. A. M a t k u r b a n o v, U. B. K h a i r u l l a e v. Constructing UAV Flight Trajectories for Agricultural Land Monitoring. – Problems of Informatics, Vol. **1**, 2025, No 66.
28. H o u g a r d y, S., F. Z a i s e r, X. Z h o n g. The Approximation Ratio of the 2-OPT Heuristic for the Metric Traveling Salesman Problem. – In: Operations Research Letters, 2020.
29. K e u n g, G. Y., et al. The Target Tracking in Mobile Sensor Networks. – In: Proc. of 2011 IEEE Global Telecommunications Conference (GLOBECOM'11), IEEE, 2011, pp. 1-5.
30. C h a v a n, P., et al. Dual-Step Hybrid Mechanism for Energy Efficiency Maximization in Wireless Network. – Cybernetics and Information Technologies, Vol. **23**, 2023, No 3, pp. 70-88.
31. K o c k e n, K., B. Ö z k ö k, H. K ö ç k e n. A Fuzzy Programming-Based Approach to a Multi-Objective Multi-Echelon Green Closed-Loop Supply Chain Problem. – Cybernetics and Information Technologies, Vol. **23**, 2023, No 3, pp. 40-55.
32. Z h a n g, L., et al. Multi-UAV Data Collection and Path Planning Method for Large-Scale Terminal Access. – Sensors, Vol. **23**, 2023, No 20, 8601.
33. N g u y e n, M. T., et al. UAV-Assisted Data Collection in Wireless Sensor Networks: A Comprehensive Survey. – Electronics, Vol. **10**, 2021, No 21, 2603.
34. K a r e g a r, P. A., D. Z. A l-H a m i d, P. H. J. C h o n g. Deep Reinforcement Learning for UAV-Based SDWSN Data Collection – Future Internet, Vol. **16**, 2024, No 11, p. 398.
35. L u o, J., Y. T i a n, Z. W a n g. Research on Unmanned Aerial Vehicle Path Planning. – Drones, Vol. **8**, 2024, No 2, p. 51.
36. X i a n g, Z., et al. Unmanned-Aerial-Vehicle Trajectory Planning for Reliable Edge Data Collection in Complex Environments. – Biomimetics, Vol. **10**, 2025, No 2, p. 109.

37. K h a n, M. A., F. F a r o o q. A Comprehensive Survey on UAV-Based Data Gathering Techniques in Wireless Sensor Networks. – ICCK Transactions on Intelligent Systems, Vol. **2**, 2025, No 1, pp. 66-75.
38. K a r e g a r, P. A., D. Z. A l-H a m i d, P. H. J. C h o n g. UAV-Enabled Software Defined Data Collection from an Adaptive WSN. – Wireless Networks, Vol. **31**, 2025, No 1, pp. 69-90.
39. Z h a n, C., Y. Z e n g, R. Z h a n g. Energy-Efficient Data Collection in UAV-Enabled Wireless Sensor Network. – IEEE Wireless Communications Letters, Vol. **7**, 2017, No 3, pp. 328-331.
40. X i a o, B., et al. Algorithms for Disk Covering Problems with the Most Points. – In: Proc. of IASTED Int'l Conf. Parallel and Distributed Computing and Systems, 2003, pp. 541-546.
41. T o k e k a r, P., et al. Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture. – IEEE Transactions on Robotics, Vol. **32**, 2016, No 6, pp. 1498-1511.
42. C a r r a b s, F., et al. An Adaptive Heuristic Approach to Compute Upper and Lower Bounds for the Close-Enough Traveling Salesman Problem. – INFORMS Journal on Computing, Vol. **32**, 2020, No 4, pp. 1030-1048.

Fast-track. Received: 10.08.2025. Revised Version: 28.08.2025. Accepted: 04.09.2025