Contents

A. Derivation of point C coordinates

Given points $A(X_A, Y_A)$, $B(X_B, Y_B)$, and length L, we derive the coordinates $C(X_C, Y_C)$ lying on AB's perpendicular bisector with $AC = L/2$ and $CB = L/2$.

First, compute the midpoint D and direction vectors:

$$D\left(\frac{X_A + X_B}{2}, \frac{Y_A + Y_B}{2}\right), \quad \vec{AB} = (X_B - X_A, Y_B - Y_A), \quad \vec{n} = (-(Y_B - Y_A), X_B - X_A) \tag{A1}$$

The unit normal vector is:

$$\hat{n} = \left(\frac{-(Y_B - Y_A)}{d_{AB}}, \frac{X_B - X_A}{d_{AB}}\right), \quad \text{where } d_{AB} = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2} \tag{A2}$$

Parametrize C along the bisector:

$$C = D + t\hat{n} \Rightarrow \begin{cases} X_C = \frac{X_A+X_B}{2} + t\frac{-(Y_B-Y_A)}{d_{AB}} \\ Y_C = \frac{Y_A+Y_B}{2} + t\frac{X_B-X_A}{d_{AB}} \end{cases} \tag{A3}$$

Apply the length constraint $AC = L/2$:

$$\sqrt{\left(\frac{X_B - X_A}{2} - t\frac{Y_B - Y_A}{d_{AB}}\right)^2 + \left(\frac{Y_B - Y_A}{2} + t\frac{X_B - X_A}{d_{AB}}\right)^2} = \frac{L}{2} \tag{A4}$$

Squaring and simplifying:

$$\frac{d_{AB}^2}{4} + t^2 = \frac{L^2}{4} \Rightarrow t = \pm\frac{\sqrt{L^2 - d_{AB}^2}}{2} \tag{A5}$$

Substituting back yields the final coordinates:

$$X_C = \frac{X_A + X_B}{2} \mp \frac{\sqrt{L^2 - d_{AB}^2}}{2}\frac{Y_B - Y_A}{d_{AB}} \tag{A6}$$

$$Y_C = \frac{Y_A + Y_B}{2} \pm \frac{\sqrt{L^2 - d_{AB}^2}}{2}\frac{X_B - X_A}{d_{AB}} \tag{A7}$$

B. Mathematical formulation

To formulate the problem, we define the binary decision variable x_{ij}^k, where $x_{ij}^k = 1$ if drone k traverses road link (i, j) for assessment, and 0 otherwise. Additionally, we introduce the continuous flow variable $f_{ij}^k \geq 0$ representing the flow carried by drone k on link (i, j). Following the notation and network transformation introduced in Sections 3 and 4, the drone-based road network assessment model is:

$$\max \quad \sum_{k\in K} \sum_{p\in \mathcal{P}} c_p \left(\sum_{j:(p,j)\in \bar{A}} x_{pj}^k \right) \tag{B1}$$

$$\textbf{s.t.} \quad \sum_{k\in K} \sum_{j:(p,j)\in \bar{A}} x_{pj}^k \leq 1 \quad \forall p \in \mathcal{P} \tag{B2}$$

$$\sum_{j\in \bar{N}} x_{ji}^k = \sum_{j\in \bar{N}} x_{ij}^k \quad \forall i \in \bar{N}, \forall k \in K \tag{B3}$$

$$\sum_{j\in \bar{N}} x_{oj}^k = \sum_{i\in \bar{N}} x_{io}^k = 1 \quad \forall k \in K \tag{B4}$$

$$x_{pj}^k + x_{jp}^k \leq 1 \quad \forall j \in \bar{N}, \forall p \in \mathcal{P}, \forall k \in K \tag{B5}$$

$$\sum_{(i,j)\in \bar{A}} t_{ij} x_{ij}^k \leq Q \quad \forall k \in K \tag{B6}$$

$$\max_{k\in K} \left\{ \sum_{(i,j)\in \bar{A}} t_{ij} x_{ij}^k \right\} \leq p_{\max} \tag{B7}$$

- Equation (B1) maximizes the total information value collected by all drones, summing c_p for each transformed artificial node traversed by drone k.
- Constraints (B2) restrict each information-bearing artificial node to at most one drone visit, preventing redundant assessments by avoiding overlapping visits from multiple drones. Notably, this constraint does not impose a single-visit requirement on nodes within the original road network G, as illustrated in Figure 3.
- Constraints (B3) enforce flow conservation for each node i and drone k, ensuring equal entry and exit counts to maintain path continuity.
- Constraints (B4) require each drone k to start and end at depot o, with exactly one exit from and one entry to o.
- Constraints (B5) forbid a drone from simultaneously using both links (p, j) and (j, p) between an artificial node p and an adjacent node j. Since each artificial node $p \in \mathcal{P}$ represents a road link in the original network, a feasible traversal should enter p from one junction and leave towards another junction. Such a pattern is an artifact of the link-to-node transformation. It does not correspond to any meaningful movement in the original road network and is therefore excluded.
- Constraints (B6) cap drone k's total flight time at battery limit Q.
- Constraint (B7) ensures the maximum flight time across all drones does not exceed $p_{\max}$, enforcing timely completion of the entire assessment.

Since the transformed network allows revisiting original nodes while requiring all assessed links to form connected routes from the depot, classical Miller–Tucker–Zemlin (MTZ) subtour elimination constraints used in the existing OP literature (Kobeaga et al., 2018; Zhang et al., 2023) are unsuitable, because they eliminate all cycles, including those that naturally occur on original nodes (see Figure 5). To simultaneously allow such revisits and prevent disconnected subtours, we propose a single-commodity flow formulation. In this formulation, (i) the depot acts as the unique supply node that provides a total amount of flow equal to the number of artificial nodes visited by a drone; (ii) each visited artificial node consumes exactly one unit of flow, representing the completion of a damage-assessment task; and (iii) original nodes serve as pure transshipment nodes with zero net flow, allowing drones to revisit them without violating flow balance. The following formulation (B8)–(B11) eliminates disconnected subtours while permitting loops at original nodes through the differential treatment of artificial and original nodes in flow conservation constraints, addressing the key modeling challenge that traditional MTZ constraints cannot accommodate.

$$\sum_{j:(o,j)\in\bar{A}} f_{oj}^k - \sum_{j:(j,o)\in\bar{A}} f_{jo}^k = \sum_{p\in\mathcal{P}} \sum_{j:(p,j)\in\bar{A}} x_{pj}^k \quad \forall k \in K \tag{B8}$$

$$\sum_{j:(j,i)\in\bar{A}} f_{ji}^k - \sum_{j:(i,j)\in\bar{A}} f_{ij}^k = \sum_{j:(i,j)\in\bar{A}} x_{ij}^k \quad \forall i \in \mathcal{P}, \forall k \in K \tag{B9}$$

$$\sum_{j:(j,i)\in\bar{A}} f_{ji}^k - \sum_{j:(i,j)\in\bar{A}} f_{ij}^k = 0 \quad \forall i \in N \setminus \{o\}, \forall k \in K \tag{B10}$$

$$f_{ij}^k \le |\mathcal{P}| \cdot x_{ij}^k \quad \forall (i,j) \in \bar{A}, \forall k \in K \tag{B11}$$

$$x_{ij}^k \in \{0,1\} \quad \forall (i,j) \in \bar{A}, \forall k \in K \tag{B12}$$

$$f_{ij}^k \ge 0 \quad \forall (i,j) \in \bar{A}, \forall k \in K \tag{B13}$$

- Constraint (B8) establishes the depot as a supply node, with net outflow equal to the number of artificial nodes visited by drone k.

- Constraint (B9) ensures each visited artificial node consumes one unit of flow, where inflow exceeds outflow by exactly one unit, representing the damage assessment task completion.

- Constraint (B10) enforces flow conservation at original nodes, where inflow equals outflow, enabling node revisits while maintaining path connectivity. This is the key mechanism that allows drones to revisit original nodes as shown in Figure 5.

- Constraint (B11) links flow to link usage, ensuring flow is only allowed on links selected by the routing decision.

- Equations (B12) and (B13) define the domains of decision variables: $x_{ij}^k \in \{0,1\}$ is a binary indicator for whether drone k traverses link (i,j), and $f_{ij}^k \ge 0$ is the continuous flow variable on link (i,j) for drone k.

C. Traditional optimization methods

In Appendix B, we formulated a mathematical model for the drone-based road damage assessment problem, which can be solved using commercial solvers such as Gurobi. To enable a more detailed performance comparison and demonstrate the advantages of the proposed AEDM over traditional optimization methods, we designed a two-phase heuristic algorithm based on established heuristic frameworks for the orienteering problem (OP) (Kobeaga et al., 2018; Yang et al., 2025). This heuristic consists of a greedy construction phase followed by a local search improvement phase. The construction phase uses a classic profit-to-time ratio rule to generate initial solutions, while the improvement phase applies relocate, exchange, and remove-insert operators that are carefully adapted to the link-to-node transformed network structure. Unlike the proposed AEDM, however, this heuristic requires hand-crafted domain knowledge, illustrating the typical limitations of traditional algorithmic approaches in newly emerging problem settings.

The construction phase builds an initial feasible solution by sequentially assigning routes to each of the K drones. For each drone k, starting from the depot o, the algorithm iteratively selects the next artificial node $p \in \mathcal{P} \setminus \mathcal{V}$ (where $\mathcal{V}$ denotes the set of already visited artificial nodes) that maximizes the profit-to-time ratio:

$$\text{score}(p) = \frac{c_p}{\text{ShortestPathTime}(u \to p)} \tag{C1}$$

where u is the current node and c_p is the information value of artificial node p. The selection is subject to the following constraints: (i) a connectivity constraint requiring that the path $u \to p$ be feasible in the transformed network $\bar{G}$, ensuring that the next node is reachable from the current position; (ii) time feasibility, requiring that the accumulated flight time plus the time to reach p and return to the depot does not exceed the time limit $\min\{Q, p_{\max}\}$; and (iii) exclusivity constraint $p \notin \mathcal{V}$, preventing redundant assessment of the same road link. Once a node is selected, the shortest path from u to p is computed and appended to the current drone's route. This process continues until no more feasible nodes can be added, at which point the drone returns to the depot and the next drone begins its route construction. The greedy construction phase terminates when all K drones have been assigned routes or all artificial nodes with positive information values have been visited.

After obtaining an initial solution from construction phase, the local search phase attempts to improve solution quality through iterative neighborhood exploration. Given the unique structure of the transformed network, we adapt three local search operators to work with triplet structures. A triplet (u_p, p, v_p) represents an artificial node p along with its predecessor u_p and successor v_p in a drone's route. The three operators are defined as follows:

- Relocate: Reverse the direction of a triplet within the same route by changing (u_p, p, v_p) to (v_p, p, u_p). This operator explores alternative approach directions to the same road link.
- Remove-insert: Remove a triplet (u_p, p, v_p) from its route and insert a different triplet (u_q, q, v_q) at a new position. This operator removes an existing assessment task and replaces it with an alternative one, potentially improving the route's total information value.

- Exchange: Swap two triplets (u_p, p, v_p) and (u_q, q, v_q) between two different drone routes, potentially improving the overall solution by reassigning assessment tasks.

The local search procedure iteratively evaluates all possible moves using these three operators. At each iteration, if a feasible neighboring solution with higher total information value is found, the current solution is updated. The search continues for a maximum of max_iterations $= 1,000$ iterations. The final solution Π^* with the highest reward R^* encountered during the search is returned as the output of the heuristic algorithm.

Algorithm 1 Greedy Heuristic with Local Search

```
Input: Transformed network $\bar{G} = (\bar{N}, \bar{A})$, flight times $t_{ij}$, information values $c_p$ for $p \in \mathcal{P}$, depot $o$, number of drones $K$, battery flight time limit $Q$, maximum allowable assessment time $p_{\max}$, maximum local-search iterations $max\_iterations$
Phase 1: Greedy construction of initial solution
Build adjacency structure from $\bar{A}$
Initialize visited artificial nodes: $\mathcal{V} \leftarrow \emptyset$
Initialize solution: $\Pi \leftarrow \emptyset$, total reward $R \leftarrow 0$
Initialize global time limit $T_{\max} = \min\{Q, p_{\max}\}$
for $k = 1$ to $K$ do
    Construct greedy route $\pi^k$:
        Set current node $u \leftarrow o$
        Iteratively select next artificial node $p \in \mathcal{P} \setminus \mathcal{V}$ maximizing
            $\text{score}(p) = \dfrac{c_p}{\text{ShortestPathTime}(u \to p)}$
        Subject to: (i) feasibility of the path $u \to p$ in the transformed network $\bar{G}$, (ii) accumulated time $\le T_{\max}$, (iii)
    feasibility of returning to $d$, (iv) $p \notin \mathcal{V}$
        Append shortest-path segments to construct complete route $\pi^k$
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{p\}$
    Add $\pi^k$ into $\Pi$; update $R$
end for
Phase 2: Local search improvement
Initialize best solution: $\Pi^* \leftarrow \Pi$, $R^* \leftarrow R$
    Define local operators:
(a) Relocate: Change a triplet $(u_p, p, v_p)$ into $(v_p, p, u_p)$ within the same route.
(b) Remove-insert: Remove triplet $(u_p, p, v_p)$ from its route and insert $(u_q, q, v_q)$ at a new position.
(c) Exchange: Swap two triplets $(u_p, p, v_p)$ and $(u_q, q, v_q)$ between routes.
for $iteration = 1$ to $max\_iterations$ do
    for operator $\in$ {Relocate, Remove-insert, Exchange} do
        Generate candidate solution $\Pi'$ using operator
        if $\Pi'$ feasible and $R(\Pi') > R(\Pi)$ then
            $\Pi \leftarrow \Pi'$
            if $R(\Pi') > R^*$ then
                $\Pi^* \leftarrow \Pi', \quad R^* \leftarrow R(\Pi')$
            end if
            break
        end if
    end for
end for
Output: Optimized routes $\Pi^*$ and reward $R^*$
```

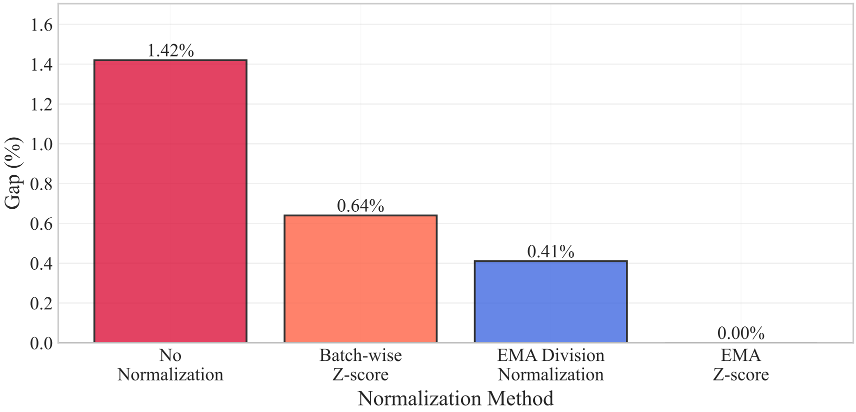

Figure D1: Performance Gap of AEDM Under Diverse Reward Normalization Methods

The complete pseudocode for the two-phase heuristic is provided in Algorithm 1. It is important to note that while this heuristic can produce reasonable solutions, its design required substantial domain expertise to: (1) adapt the profit-to-time ratio heuristic to the transformed network structure, (2) develop triplet-based operators that respect the exclusivity constraints of artificial nodes while allowing revisits to original nodes, and (3) carefully tune the search parameters and stopping criteria. This design complexity highlights a key advantage of the proposed AEDM: it eliminates the need for such hand-crafted problem-specific heuristics by learning effective routing policies directly from data.

D. Ablation study

To evaluate the impact of the proposed reward normalization strategy (EMA with Z-score normalization) on model performance, we conducted an ablation study by comparing four normalization variants of the AEDM framework:

a) EMA with Z-score normalization: Combines EMA of both mean and variance to compute Z-score normalized rewards, as proposed in Equations (15)–(17).

b) Batch-wise Z-score normalization: Normalizes rewards within each batch using the batch mean and variance, without exponential smoothing.

c) EMA mean division normalization: Normalizes rewards by dividing by the EMA of the mean (ignoring variance).

d) No normalization: Uses raw rewards without any regularization.

All variants were trained under identical conditions detailed in Section 6.1 and evaluated on 400-node networks with $p_{\text{max}} = 45$ minutes and 4, 5, 6, 7 drones (i.e., four scenarios). Performance was quantified by the average relative gap across these four scenarios, defined as Gap $= (y - y_{\text{other}})/y$. Here, y denotes the objective value from the EMA with Z-score normalization, and y_{other} represents that from other normalization variants. As visualized in Figure D1,

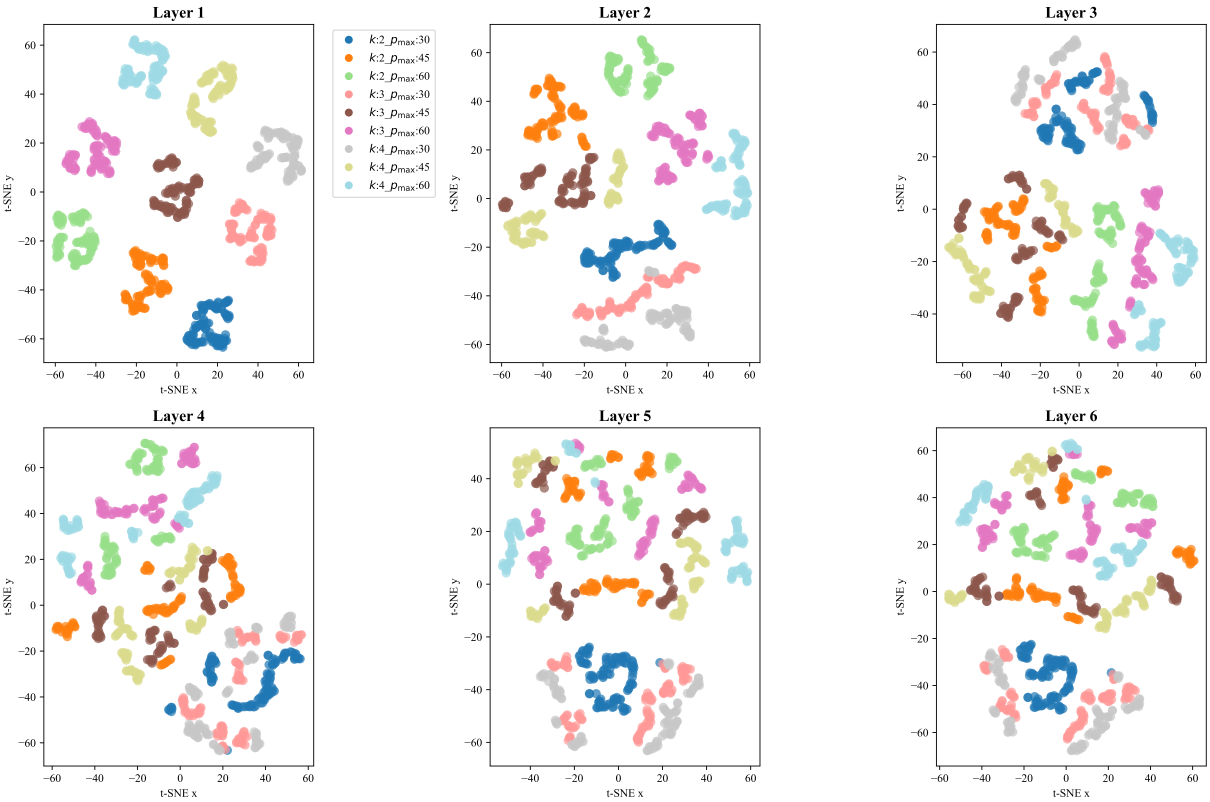

Figure E1: Method Interpretability: Encoder Latent Space via Layer-Wise t-SNE Analysis

the proposed reward normalization strategy achieves the best performance, whereas the "No normalization" variant yields the worst. These results underscore the critical role of EMA-based mean and variance tracking in stabilizing reinforcement learning for multi-task drone routing. The proposed normalization strategy ensures reward comparability across diverse parameter regimes (e.g., varying drone numbers and time limits), enabling effective model generalization to unseen scenarios.

E. Method interpretability

To explore the model's interpretability, we employ t-distributed stochastic neighbor embedding (t-SNE) to project high-dimensional representations, which are learned under varying drone numbers k, $p_{\max}$, and a fixed 100-node scale, into a 2D space. With 100 instances per parameter combination, Figure E1 visualizes the evolution of features across the output of each encoder layer.

In Layer 1, features cluster distinctly by k-$p_{\max}$ combinations, indicating that the model first learns superficial parameter distinctions in initial layers and validating its ability to recognize input configurations. Progressing to Layer 2, clusters remain distinct but exhibit early fusion, where features begin interacting across k and $p_{\max}$ while retaining core differences. In Layers 3 and 4, clusters further soften and then mix: distributions of different $k/p_{\max}$ values interweave, reducing strict separation. In Layers 5 and 6, clusters dissolve into complex, interwoven distributions.

This layer-by-layer t-SNE evolution reveals a clear learning trajectory: initial layers (1–2) recognize and separate raw input parameters (k, $p_{\max}$); middle layers (3–4) fuse these features; final layers (5–6) abstract core problem structures, prioritizing solution logics over input specifics. This confirms the model learns generalizable principles (not merely parameter patterns), leveraging the transformer architecture to distill raw inputs into abstract representations of the core problem. The cross-layer feature fusion validates that the model transitions from "memorizing" to "reasoning", which is a critical hallmark of effective problem-solving in complex tasks.

References

Kobeaga, G., Merino, M., Lozano, J.A., 2018. An efficient evolutionary algorithm for the orienteering problem. Computers & Operations Research 90, 42–59.

Yang, X., Zhang, L., Qian, H., Song, L., Bian, J., 2025. Heuragenix: Leveraging llms for solving complex combinatorial optimization challenges. arXiv preprint arXiv:2506.15196 .

Zhang, G., Jia, N., Zhu, N., Adulyasak, Y., Ma, S., 2023. Robust drone selective routing in humanitarian transportation network assessment. European Journal of Operational Research 305, 400–428.