2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

An Attention-aware Deep Reinforcement Learning Framework for UAV-UGV Collaborative Route Planning

Md Safwan Mondal[1,†], Subramanian Ramasamy[1], James D. Humann[2], James M. Dotterweich[3], Jean-Paul F. Reddinger[3], Marshal A. Childers[3], Pranav Bhounsule[1]

***Abstract*— Unmanned aerial vehicles (UAVs) possess the capability to survey vast areas, yet their operational range is limited by their battery capacity. Deploying mobile recharging stations via unmanned ground vehicles (UGVs) can significantly enhance the endurance and effectiveness of UAVs. However, optimizing the routes for both UAVs and UGVs, referred to as the UAV-UGV cooperative routing problem, requires a sophisticated planning framework to determine the vehicles' routes and their recharging points. To address this, in this paper, we utilize a deep reinforcement learning (DRL) based framework equipped with multi-head attention layers. The framework is designed to sequentially select actions to construct routes for the UAV and UGV and to establish their rendezvous points for recharging. We evaluate our framework across various problem instance sizes and distributions, comparing it against recent heuristic-based methods and an existing learning-based method as baselines. Our proposed algorithm surpasses these baselines in terms of solution quality and runtime efficiency in the test scenarios, thus proving its effectiveness. Additionally, we investigate the application of our DRL policy in online mission planning to accommodate dynamic changes within the mission scenario.**

I. Introduction

Unmanned aerial vehicles (UAVs) have rapidly evolved as an emerging technology, finding profound applications in both military and civilian sectors [1]–[3]. They are critical for real-time sensing in scenarios like traffic monitoring [3], border security [4], disaster management [5] and forest fire surveillance [6], all of which require continuous operation. However, a major limitation of UAVs in such persistent applications is their restricted flight time due to limited battery capacity. This challenge can be mitigated by leveraging the synergies of multi-agent systems that combine UAVs with unmanned ground vehicles (UGVs), which can act as mobile stations to facilitate recharging for UAVs. This collaborative approach can enhance the overall task efficiency and prolong the UAVs' operational longevity [7].

In this work, we address a ***cooperative routing problem*** involving a team of a UAV and UGV, tasked with visiting a set of designated mission points in the shortest possible time (see Fig. 1). The UAV operates under a limited battery life and is supported by the UGV, which acts as a mobile recharging depot. The UGV is also limited by its speed and can only travel on the road network. Given the heterogeneity of the vehicles, the challenge lies in strategizing their routes to ensure that the UGV can effectively recharge the UAV to execute the mission optimally. This necessitates a comprehensive cooperative routing framework that optimally plans the routes for both the UAV and UGV while synchronizing their rendezvous for recharging.

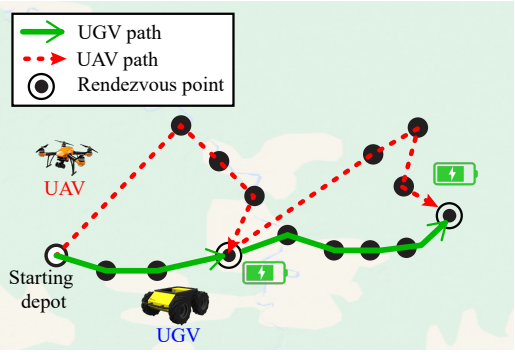

Fig. 1: Illustration of the fuel-constrained UAV-UGV cooperative routing problem: The UAV visits a set of mission points and lands on the UGV to recharge. The goal is to plan the routes for both the UGV and UAV to minimize the time taken to visit all the mission points while meeting the fuel and speed constraints of the UAV and UGV.

A. Related works

Extensive research has been conducted on different variants of UAV-UGV cooperative routing problem [8]–[10] across the fields of transportation and robotics. In transportation, the Truck-Drone coordinated delivery problem, analogous to the UAV-UGV cooperative routing problem, is modeled as a variant of the Vehicle Routing Problem with Drones (VRP-D) [11] or the Traveling Salesman Problem with Drones (TSP-D) [12]. These routing problems are often modeled using Mixed Integer Linear Programming (MILP) [13] and addressed by employing a multi-echelon strategy [14]. While MILP can theoretically yield an optimal solution, it suffers from the curse of dimensionality, being an NP-Hard problem, and thus becomes computationally infeasible for larger-scale problems [15]. As a result, handcrafted heuristics and metaheuristics are employed to produce suboptimal solutions within an acceptable time. However, these heuristics are mostly tailored to specific problem types, making the

[1]Md Safwan Mondal, Subramanian Ramasamy and Pranav A. Bhounsule are with the Department of Mechanical and Industrial Engineering, University of Illinois Chicago, IL, 60607 USA. mmonda4@uic.edu, sramas21@uic.edu, pranav@uic.edu [2]James D. Humann is with DEVCOM Army Research Laboratory, Los Angeles, CA, 90094 USA. james.d.humann.civ@army.mil [3]James M. Dotterweich, Jean-Paul F. Reddinger, Marshal A. Childers are with DEVCOM Army Research Laboratory, Aberdeen Proving Grounds, Aberdeen, MD 21005 USA. james.m.dotterweich.civ@army.mil, jean-paul.f.reddinger.civ@army.mil, marshal.a.childers.civ@army.mil

† Corresponding author, *This work was supported by ARO grant W911NF-14-S-003.

process labor-intensive and limiting their generalizability and effectiveness.

In recent years, learning-based methodologies have become a promising alternative for solving the vehicle routing problem, its variants, and other combinatorial optimization problems. Kool et al. [16] introduced an encoder-decoder Transformer architecture that outperformed several classical heuristic methods across a variety of routing problems. Similarly, Li et al. [17] employed a deep reinforcement learning (DRL) method with attention mechanisms to address the heterogeneous capacitated vehicle routing problem, achieving superiority in both solution quality and computational efficiency over non-learning baselines. Furthermore, Wu et al. [18] explored the truck and drone based last-mile delivery problem using reinforcement learning (RL). They decomposed the optimization problem into customer clustering and routing components, applying an encoder-decoder framework with RL to solve it effectively. On a similar note, Fan et al. [19] utilized a multi-head attention mechanism combined with a DRL policy to design routes for an energy-constrained UAV, although their model assumed a fixed set of recharging locations for the UAV. Despite these numerous RL frameworks addressing various vehicle routing problems, there remains an opportunity to explore the UAV-UGV collaborative routing problem with its underlying multi-agent aspect. In cooperative routing, it is crucial not only to decide the sequence of mission point visits but also to determine the recharging instances between agents, ensuring their successful coordination. Ramasamy et al. [20] demonstrated the impact of these recharging instances on the overall efficiency of cooperative routing, underscoring the need for effective synchronization between the vehicles. Therefore, this paper proposes an RL framework based on an encoder-decoder transformer architecture and policy gradient method to determine coordinated routing between an energy-constrained UAV and a UGV acting as a mobile recharging station. The framework has been tested across different problem sizes and distributions, indicating its generalizability. To this end, we present the following novel contributions:

1. We model the energy-constrained UAV-UGV cooperative routing problem as a Markov Decision Process (MDP), enabling its solution through RL, which utilizes an encoder-decoder based transformer architecture with attention layers.
2. We evaluate the framework across various problem sizes and distributions, showcasing its generalization capabilities.
3. We compare our method against recent heuristic methods for cooperative routing and learning-based approaches. Our approach demonstrates superior solution quality, highlighting its effectiveness and reliability.
4. We also investigate the framework's utility for online route planning in response to dynamically appearing mission points.

II. Problem Formulation

A. Problem overview

To formally define the problem, let us consider a system consisting of a fuel-constrained UAV, U^a and a UGV, U^g tasked with visiting a set of n mission points, $\mathcal{M} = \{m_0, m_1, ..., m_n\}$ spread across a scenario. We have two types of mission points: some mission points are located within a road network G, can be surveyed either by UAV flyover or by UGV based road visit (referred to as ground points, $\mathcal{M}_g$). Other mission points, situated outside the road network, are accessible only to the UAV (referred to as UAV points, $\mathcal{M}_a$); hence, mission points $\mathcal{M}$ can be defined as $\mathcal{M} = \mathcal{M}_g \cup \mathcal{M}_a$. The vehicles exhibit heterogeneous characteristics: the UAV has a limited battery capacity F^a but can fly with a faster velocity v^a, whereas the UGV moves at a slower pace v^g, and operates exclusively on the road network G. For recharging, the UAV meets the UGV at any ground point, spends a fixed recharging service time T_R there, and then resumes its flight by taking off from the UGV. The mission initiates as soon as the UAV or UGV leaves the starting depot and concludes once every mission point has been visited, finishing with the UAV completing its final recharge on the UGV. The objective is to devise a strategy for collaborative operation between the UAV and the UGV to visit all mission points in the scenario at least once in the shortest possible time. Therefore, the challenge is multifaceted, requiring optimization of the UAV and UGV route sorties and scheduling their recharging instances to synchronize in time and location.

B. MDP formulation

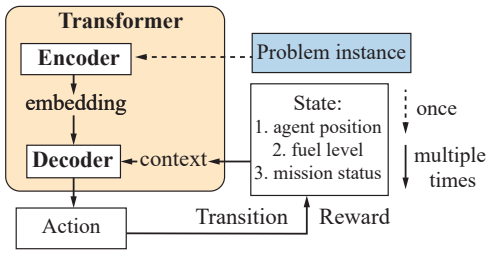

Fig. 2: MDP representation for the UAV-UGV cooperative routing problem utilizing a Transformer network

The problem can be modeled as a sequential decision-making system where the agents sequentially select the mission points (see Fig. 2). This system can be formulated as a Markov Decision Process (MDP), with its components defined by the tuple $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T} >$, as follows:
1) **State Space** $(\mathcal{S})$**:** At any decision making step, the state of the environment $s_t \in \mathcal{S}$ is defined as, $s_t = (p_t, f_t,\ q_t)$, where, $p_t = \{x_t, y_t\}$ represents the current position of the agent and its fuel level is indicated by f_t. Additionally, $q_t = \{x^i, y^i, d_t^i\}$ highlights the coordinates and mission visitation status of the task points $m_i \in \mathcal{M}$; d_t^i will be 1 if the task point m_i is already visited or 0 if not.
2) **Action Space** $(\mathcal{A})$**:** The selection of a mission point is defined as the action $a_t \in \mathcal{A}$. The agents can perform two types of actions: *visiting* (on both UAV points $\mathcal{M}_a$ and ground points $\mathcal{M}_g$) or *recharging* (only on ground points $\mathcal{M}_g$). Since $\mathcal{M}_g$ is used for both visiting and recharging, the action space is defined as, $\mathcal{A} = \{\mathcal{M}_g(\text{recharging}) + \mathcal{M}_g(\text{visiting}) + \mathcal{M}_a(\text{visiting})\}$. We mask out (details in

subsection III-A.2) infeasible actions from the action space $\mathcal{A}$ based on the current state s_t at time t.

3) **Reward ($\mathcal{R}$):** Aligning with the objective to minimize total mission completion time, we define the reward $\mathcal{R}$ as the negative of total mission time. The total mission time is calculated by summing the step elapsed time r_t at every decision making step as $\mathcal{R} = -\max_{u_a/u_g} \sum_{t=0}^{T} r_t$. Here, the step elapsed time includes travel time between the nodes and additional recharge service time in case of a *recharging* action. It can be indicated as $r_t = r(s_t, a_t) = t_{ij} + T_R$, when agent travels between node i and j at timestep t ($T_R = 0$ for *visiting* action). For effective reward shaping, we impose a substantial penalty, P, in case of a mission failure. Thus, the net reward r_t can be written be as: $\mathcal{R} = -\max_{u_a/u_g} \sum_{t=0}^{T} r_t - P$ (in the case of task failure, $P = 1000$, otherwise $P = 0$). This reward structure is designed to promote successful task completion and facilitate faster learning convergence.

4) **Transition ($\mathcal{T}$):** The transition function updates the current state s_t to the next state $s_{t+1} = (p_{t+1}, f_{t+1}, q_{t+1})$ based on the taken action a_t. The new position of the agent will be the chosen task location, $p_{t+1} = (\{x_{t+1}, y_{t+1}\} \equiv a_t)$ and the fuel level will be updated as $f_{t+1} = f_t - f_{ij}$ if *visiting* task is performed or $f_{t+1} = F^a$ for *recharging* task. The visitation status of the mission point is also updated as $d^i_{t+1} = 1$, for mission point m_i corresponds to a_t. Since stochasticity is not assumed the transition is considered to be deterministic.

III. Reinforcement learning framework

In this section, we present the encoder-decoder based transformer network with RL algorithm to learn the routing policy π_θ, with a trainable parameter θ. Starting from the initial state s_0, the policy π_θ takes action a_t to select a mission point for *visiting* or *recharging* based on the current scenario state s_t until the terminal state s_T is reached. The final solution of the policy network will be our cooperative route $\mathcal{T}$, consisting of a series of mission points chosen sequentially. It can be represented by a joint probability distribution as follows:

$$\mathbb{P}(\mathcal{T}; \theta) = \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t)\mathbb{P}(s_{t+1}|s_t, a_t) \tag{1}$$

Here, T is the timesteps till mission termination and $\mathbb{P}(s_{t+1}|s_t, a_t) = 1$, as we have chosen deterministic state transition.

A. Encoder-Decoder Transformer architecture

To learn the routing policy π_θ, we utilize an encoder-decoder transformer architecture similar to Kool et al. [16]. However, the architecture is adapted to cooperative routing problem settings, and a one-agent-per-decoding strategy [21] is followed in the decoder to accommodate multi-agent actions. The encoder translates the coordinates of the mission points into a nuanced, high-dimensional embedding, enhancing feature extraction. Subsequently, the decoder exploits these embeddings to take actions based on the **contextual** information derived from the current state of the scenario. A detailed description of the transformer architecture (see Fig. 3) is explained here:

1) Encoder

We have incorporated a multi-head attention (MHA) mechanism [22] in the encoder to achieve a higher-dimensional representation of the raw features of the problem instance. The encoder accepts the 3D vector representation of the mission points as input, $X = (o_i = \{x_i, y_i, b_i\}, \forall\ m_i \in \mathcal{A})$, where (x_i, y_i) represents the normalized coordinates and b_i is a binary variable indicating if mission points are eligible for recharging. Node inputs are linearly projected into node embedding $h_i^0 = W^0 o_i + b^0$ with dimension $d_h = 128$. This input embedding is subsequently transformed using L multi-head attention layers into an advanced input embedding h_i^L, allowing a more detailed understanding of the relationships among the task points. Within each attention layer $l \in L$, three vectors, *Query*, *Key* and *Value* are calculated from previous layer's node embedding h_i^{l-1}, with the dimension of *Query*/*Key*, *Value* set as, $d_q/d_k = d_v = \frac{d_h}{M}$, where $M = 8$ is the number of attention heads. For each head $j \in 1, 2, ...M$, the attention scores Z_j^l are calculated between *Query* and *Key*, which are concatenated together to give attention output $\text{MHA}(h_i^{l-1})$ of that layer. The calculations are shown here:

$$q^l_{i,j} = h_i^{l-1} W^l_{q,j},\ k^l_{i,j} = h_i^{l-1} W^l_{k,j},\ v^l_{i,j} = h_i^{l-1} W^l_{v,j} \tag{2}$$

$$Z_j^l = \text{softmax}\left(\frac{q^l_{i,j} {k^l_{i,j}}^T}{\sqrt{d_k}}\right) v^l_{i,j} \tag{3}$$

$$\text{MHA}(h_i^{l-1}) = \text{Concat}(Z_1^l, Z_2^l, ..., Z_j^l) \tag{4}$$

Here, $q^l_{i,j}, k^l_{i,j}$ and $v^l_{i,j}$ are the *Query*, *Key* and *Value* respectively in head j and $W^l_{q,j} \in \mathbb{R}^{d_h \times d_q}, W^l_{k,j} \in \mathbb{R}^{d_h \times d_k}$ and $W^l_{v,j} \in \mathbb{R}^{d_h \times d_v}$ are the trainable parameter matrices in the attention layer l. The attention output $\text{MHA}(h_i^{l-1})$ is followed by a feed-forward layer (FF) with ReLU activation function to give the node embedding of that layer. A residual skip connection and Batch-Normalization (BN) are applied in both MHA and FF sublayers as shown below:

$$\hat{h}_i^l = BN(h_i^{l-1} + \text{MHA}(h_i^{l-1})) \tag{5}$$

$$h_i^l = BN(\hat{h}_i^l + FF(\text{ReLU}(\hat{h}_i^l))) \tag{6}$$

After L attention layers, we obtain the final node embedding as h_i^L, which we send to the decoder for further processing.

2) Decoder:

In general, during each decision-making step the decoder determines the probability of selecting each available node as an action based on the encoder's node embedding h_i^L and a **context** vector, which provides insights into the current scenario state. In our multi-agent problem setting, we refine the one-agent-per-decoding-step strategy of the work [21] to determine the active agent for taking action. Unlike their method, which alternates agents at every decision point, our approach permits one agent to conclude a full sortie before switching. For instance, the UAV performs a series of actions, visiting multiple mission points before concluding its sortie by selecting a refueling node. Only then is the UGV chosen as the agent to execute its tasks and coordinate with the UAV at the chosen refueling point, after which the UAV is

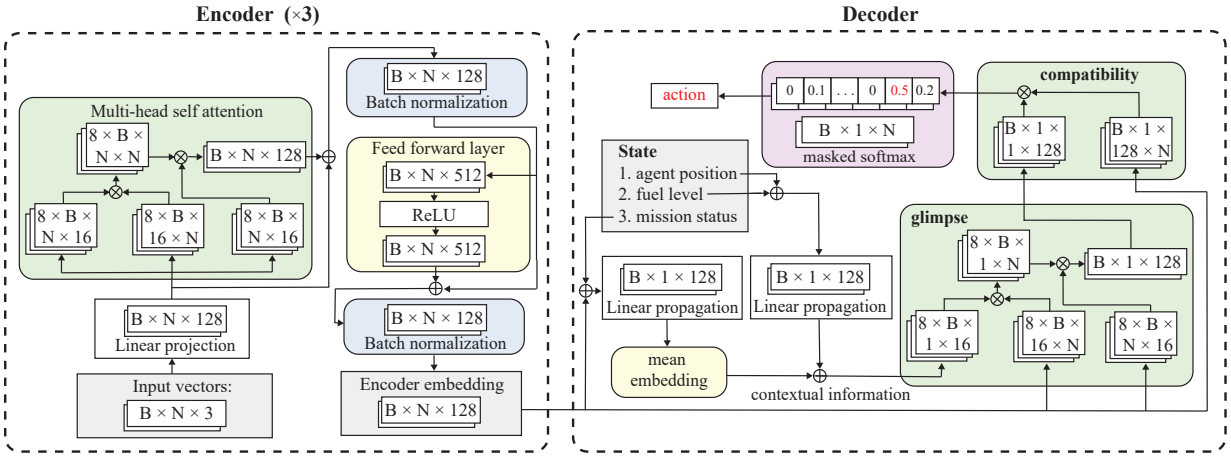

Fig. 3: Architecture of the proposed Transformer network. The encoder comprises three attention layers that generate input embeddings from raw data, while the decoder constructs a context vector based on the current state. It utilizes both the input embedding and the context vector, passing them through a multi-head attention layer and a single-head attention layer to determine the action.

selected again as the agent for its next sortie. This approach is important because the UGV's route sortie should depend on the UAV's sortie, requiring a complete view of the UAV's actions for effective planning. By allowing the UAV to finish its sortie before the UGV begins, we ensure a coherent sequence of actions and more efficient agent collaboration throughout the mission. At every decision making step t, we build the **context** vector h_t^c, utilizing the agent's current position embedding $h_{j,t}^L$, for $j \in p_t$, the current state's visit status d_t^i, the agent's fuel level f_t and the encoder node embedding h_i^L, as shown below:

$$\bar{h}_t = \frac{1}{n}\sum_{i=0}^{n}(\text{Cat}(h_i^L, d_t^i)W_g) \tag{7}$$

$$h_t^c = \bar{h}_t + \text{Cat}(h_{j,t}^L, f_t)W_c \tag{8}$$

Here, W_g, W_c are trainable parameters. Once the context vector is constructed from the current state, the decoder employs an MHA layer by taking the context vector as the *Query* and encoder node embedding as the *Key*/*Value* to calculate the glimpse h_t^g as follows:

$$h_t^g = \text{MHA}(h_t^c,\ h_i^L W_k^g,\ h_i^L W_v^g) \tag{9}$$

Here, W_k^g, W_v^g are trainable parameters. Finally, once glimpse h_t^g is obtained, it is fed as *Query* q_t and h_i^L as the *Key* k_t in a single head attention layer to compute their compatibility h_t. An important step here is to mask out infeasible actions based on the current scenario state. We consider 1) already visited mission points (for *visiting* action) 2) unreachable mission points at current fuel level (when UAV is the agent) 3) mission points that would fail the agent to reach any refuel stop (when UAV is the agent) 4) unreachable mission points within UAV's previous sortie time period (when UGV is the agent) as logical constraints to decide infeasible actions for masking. The compatibility between the *Query* q_t and *Key* k_t is calculated as shown:

$$q_t = h_t^g W_q, k_t = h_i^L W_k \tag{10}$$

$$h_t = \begin{cases} C_p \cdot \tanh\left(\frac{q_t {k_t}^T}{\sqrt{d_q}}\right) & \text{if feasible} \\ -\infty & \text{else.} \end{cases} \tag{11}$$

Here, W_q, W_k are trainable matrices and $C_p = 10$ is a clipping parameter for better exploration. Output probabilities for the actions are calculated using the softmax function as:

$$\pi_\theta(a_t|s_t) = \text{softmax}(h_t) \tag{12}$$

Following the above process, the decoder sequentially takes actions and alters the agents till the mission is terminated. We apply sampling decoding strategy in the training procedure, and both greedy and sampling strategies are incorporated during the evaluation.

B. Training method

The training algorithm (see Algorithm 1) leverages the REINFORCE policy gradient method [23]. It comprises 1) our policy network π_θ that takes action by calculating probability distribution over the actions and 2) rollout baseline network π_ϕ that has identical structure to π_θ but takes action greedily (action with maximum probability). At every training iteration, we calculate the routes and their reward for a batch of instances, and the expected baseline rewards for those instances come from greedy rollout in the baseline network (lines 4-11). We update the policy network parameter θ following the policy gradient algorithm (lines 12-13). Moreover, after each epoch, the baseline network's parameters ϕ are updated with those of the policy network if it underperforms in a paired t-test (line 15). Both the policy and baseline networks are iteratively updated during the training process to yield an optimal policy at the end of training. REINFORCE is chosen for its ability to learn directly from interactions, eliminating the need for explicit modeling, which enhances scalability in large problem sizes. Unlike some model-based methods with exponential time complexity due to tree search, REINFORCE generally maintains a linear time complexity relative to the number of epochs, steps, and gradient computations, making it a more practical and scalable option.

Algorithm 1: Policy network training using REINFORCE algorithm

Input: Policy network π_θ, Baseline network π_ϕ, epochs E, Number of batches N, batch size B, episode length T
Output: Trained policy network $\pi_{\theta'}$

for *epoch in* $1 \dots E$ **do**
  Sample N batches from dataset
  for *iteration in* $1 \dots N$ **do**
    for *instance* b *in* $1 \dots B$ **do**
      Initialize $s_{0,b}$ at $t = 0$
      while $t < T$ **do**
        Get action $a_{t,b} \sim \pi_\theta(a_{t,b}|s_{t,b})$
        Obtain reward $r_{t,b}$ and $s_{t+1,b}$
        $t = t + 1$
      $\mathcal{R}_b = -\max_{u_a/u_g} \{\sum_{t=0}^{T} \boldsymbol{r}_{t,b}\}$
      Baseline reward $\mathcal{R}_b^\phi$ from greedy rollout with π_ϕ
    Compute gradient:
$$J \leftarrow \frac{1}{B} \sum_{b=1}^{B} (\mathcal{R}_b - \mathcal{R}_b^\phi) \nabla_\theta \log \pi_\theta \left(s_{T,b} \mid s_{0,b}\right)$$
    Update $\theta \leftarrow \theta + \alpha \nabla_\theta J$
  if *OneSidedPairedTTest*$(\pi_\theta, \pi_\phi) < 0.05$ **then**
    $\phi \leftarrow \theta$

IV. RESULTS

A. Dataset details

The effectiveness of our proposed algorithm has been evaluated through extensive computational experiments. We have simulated the fuel-constrained UAV-UGV cooperative routing problem over an area of 20 km $\times$ 20 km, employing a single UAV-UGV system as outlined in the problem statement. The UAV and UGV move at speeds of $v^a = 10$ m/s and $v^g = 4.5$ m/s, respectively. The UAV has a fuel capacity of $F^a = 287.7$ kJ and follows a fuel consumption profile defined as, $\mathcal{P}^a = 0.0461(v^a)^3 - 0.5834(v^a)^2 - 1.8761v^a + 229.6$ (modeled after [24]). The starting depot and the UAV mission points $\mathcal{M}_a$ are uniformly sampled around the road network points $\mathcal{M}_g$, considering the UAV's fuel radius. For computational efficiency, we have kept UGV road network G fixed, but chosen random road network points $\mathcal{M}_g$ from it that act as recharging stops (accessible to both UAV and UGV). We train our model on three different problem sizes: 15 UAV points with 5 ground points (**U15G5**), 30 UAV points with 10 ground points (**U30G10**), and 45 UAV points with 15 ground points (**U45G15**). For each problem size, training is conducted on a total of 5120000 instances, with 256 instances per batch across 200 batches over 100 epochs. The instances are generated on-the-fly during training. We employ the Adam optimizer with a learning rate of 10^{-4} and a decay rate of 0.995, which is adjusted at the conclusion of each epoch. The training is completed on a server equipped with an RTX 2080 Ti GPU. The average training time per epoch is approximately 3.00 minutes for U15G5, 5.00 minutes for U30G10, and 7.00 minutes for U45G15. Our model's effectiveness is evaluated on two different distributions of mission points and the results are compared against recent state-of-the-art heuristic methods. Additionally, we test the model's generalization capability on larger problem sizes and extended road network. A simulated case study on an actual task site is also conducted to explore the practical application of our model in dynamic planning.

B. Comparison evaluation

Given the complexity and specificity of the problem, there are no standard benchmarks available, nor can an exact solution be easily discerned, as the problem becomes intractable with an increasing number of mission points. In cooperative routing problems for heterogeneous vehicles, a prevalent strategy is to employ a multi-level optimization or multi-echelon approach (e.g.,'UGV first, UAV second', 'UAV first, UGV second','Truck first, drone second' etc.) [25]–[28]. This methodology simplifies the problem by dividing it into more manageable subproblems and solving them using heuristics. We adopt this approach to design our baseline methods. Initially, we determine the UGV route by solving the minimum set cover problem to identify refuel stop locations and by solving the traveling salesman problem to connect these refueling stops, thus establishing the UGV path. Based on this UGV path, the UAV route is derived by modeling it as an energy-constrained vehicle routing problem with time window constraints (EVRPTW) and solving it using constraint programming with the following metaheuristics: 1) **Guided Local Search (GLS)**, 2) **Tabu Search (TS)**, and 3) **Simulated Annealing (SA)** in the Google OR-Tools™ CP-SAT solver [29]. The details about the implementation of this methodology and its effectiveness are discussed in our previous works [30], [31]. In the evaluation problem instances, ground points are uniformly sampled from the road network. However, instead of uniformly sampling the UAV mission points, we opt for two distributions: a) Gaussian distribution and b) Rayleigh distribution, both centered around the road network points. Following Kool et al. [16], we incorporate two types of decoding strategies in our deep reinforcement learning framework: namely, greedy decoding, where actions with the maximum probability are chosen at every decision-making step, and sampling decoding, where $\mathbb{N}$ trajectories are sampled and the best solution is selected from them. We set $\mathbb{N}$ to 1024 (DRL(1024)) and 10240 (DRL(10240)) in our evaluation instances. For a learning-based baseline, we adapt the Attention Model (AM) [16] for our problem settings with same decoding strategies. For the evaluation process, we utilize an Nvidia Quadro P2200 GPU. Given that the heuristic approach demands more computational time, we evaluate 100 test instances, and all computations are implemented in Python.

Table I lists the average objective values and runtimes for problem instances across three problem sizes, comparing all methodologies. Ideally, a lower objective value achieved in a shorter runtime signifies superior solution quality. The table also shows the optimality gap for each method, calculated as the difference between their objective values and the best objective function value found, as shown here:

$$\text{optimality gap} = \frac{\text{Obj.} - \text{Obj.}_{\text{best}}}{\text{Obj.}_{\text{best}}} \times 100\% \qquad (13)$$

The table reveals that the proposed DRL policy achieves

TABLE I: Comparison evaluation of the DRL policy across problem sizes and distributions.

	Method	U15G5			U30G10			U45G15		
		Obj. (min.)	Gap (%)	Time (sec)	Obj. (min.)	Gap (%)	Time (sec)	Obj. (min.)	Gap (%)	Time (sec)
Gaussian Distribution	GLS	247	30.0	154.0	329	28.5	168.0	405	28.6	191.0
	TS	247	30.0	155.0	331	29.3	169.0	406	28.9	191.0
	SA	250	31.6	155.0	335	30.9	169.0	412	30.8	192.0
	AM(greedy)	224	17.9	**0.5**	297	16.0	**1.0**	371	17.8	**1.3**
	AM(1024)	196	3.2	1.8	266	3.9	3.2	328	4.1	4.8
	AM(10240)	191	0.5	19.1	261	2.0	33.1	323	2.5	48.9
	DRL(greedy)	220	15.8	0.7	294	14.8	1.2	360	14.3	1.4
	DRL(1024)	192	1.1	2.0	261	2.0	3.8	322	2.2	4.9
	DRL(10240)	**190**	0.0	20.6	**256**	0.0	37.6	**315**	0.0	49.6
Rayleigh Distribution	GLS	281	26.0	155.0	382	26.5	169.0	444	22.0	190.0
	TS	281	26.0	156.0	382	26.5	169.0	445	22.3	191.0
	SA	282	26.5	155.0	384	27.2	169.0	454	24.7	192.0
	AM(greedy)	249	11.7	**0.6**	354	17.2	1.1	424	16.5	3.7
	AM(1024)	227	1.8	2.1	313	3.6	3.9	379	4.1	6.8
	AM(10240)	**223**	0.0	21.0	308	2.0	38.5	373	2.5	60.7
	DRL(greedy)	251	12.6	0.8	345	14.2	**1.0**	423	16.2	**1.7**
	DRL(1024)	227	1.8	2.2	307	1.7	3.8	368	1.1	5.0
	DRL(10240)	**223**	0.0	21.6	**302**	0.0	37.0	**364**	0.0	50.3

the **minimum** mission time across all problem sizes and distributions compared to the baselines. While the AM baseline generates solutions comparable to those of the DRL method, the optimality gap between AM and DRL algorithms widens as problem sizes increase; for AM(10240) versus DRL(10240), the gap ranges from 0-0.5% in smaller problem sizes (U15G5) to 2% in medium sizes (U30G10), and 2.5% in larger problem sizes (U45G15). This underscores the DRL policy's efficiency in managing larger scenarios. Within the DRL framework, the sampling decoding methods outperform the greedy approach, exhibiting an optimality gap of 14-16%, at the cost of higher computational time as DRL(10240) is 30 times, and DRL(1024) is 3 times slower than the DRL(greedy) method. Generally, solution time increases with growing problem sizes, with the greedy decoding strategy providing the quickest runtime for both AM and DRL methods. Notably, DRL(1024) emerges as the optimal choice, securing an optimality gap of less than $\sim$2.2% compared to DRL(10240), but with a computation time tenfold faster, representing a favorable trade-off between solution quality and runtime efficiency. Typically, the Rayleigh distribution results in 15-17% longer mission periods than the Gaussian distribution, yet the comparative performance of the methodologies remains consistent across both distributions. The animations of the routes obtained from different methods across various scenarios can be viewed at `http://tiny.cc/hmnjxz`.

C. Generalization

To assess the generalization capability of the proposed DRL framework, we examine different testing instances by modifying two aspects of the problem scenarios: 1) increasing the number of mission points and 2) extending the road network. We generate 20 test instances each for configurations with 60 UAV points and 20 ground points (**U60G20**), 75 UAV points, and 25 ground points (**U75G25**), employing Gaussian and Rayleigh distributions. We apply the learned policy from the U45G15 model to these new testing instances to evaluate its effectiveness. Additionally, we compare the model's performance against the baseline methods, as listed in Table II.

According to Table II, within the Gaussian distribution,

TABLE II: Performance across larger scenarios

	Method	U60G20			U75G25		
		Obj. (min.)	Gap (%)	Time (sec)	Obj. (min.)	Gap (%)	Time (sec)
Gaussian Distribution	GLS	438	16.8	245.0	530	19.1	248.0
	TS	442	17.9	243.0	527	18.4	249.0
	SA	459	22.4	245.0	535	20.2	249.0
	AM(greedy)	449	19.7	4.2	520	16.9	2.5
	AM(1024)	391	4.3	11.9	464	4.3	12.5
	AM(10240)	384	2.4	110.8	457	2.7	137.7
	DRL(greedy)	428	14.1	**3.8**	509	14.4	**2.1**
	DRL(1024)	382	1.9	9.3	455	2.2	10.6
	DRL(10240)	**375**	0.0	116.0	**445**	0.0	132.6
Rayleigh Distribution	GLS	522	21.1	245.0	609	21.8	250.0
	TS	524	21.6	243.0	613	22.6	248.0
	SA	537	24.6	243.0	621	24.2	250.0
	AM(greedy)	524	21.6	3.6	597	19.4	**2.4**
	AM(1024)	452	4.9	14.9	519	3.8	10.1
	AM(10240)	439	1.9	120.5	512	2.4	139.2
	DRL(greedy)	523	21.3	**3.3**	581	16.2	2.5
	DRL(1024)	440	2.1	13.7	514	2.8	14.5
	DRL(10240)	**431**	0.0	130.0	**500**	0.0	141.0

learning-based methods with sampling decoding strategy, outperform heuristic methods for both the U60G20 and U75G25 problem sizes, as they produce lower objective values. The DRL policy generalizes better compared to the AM policy, producing better solution quality (with a 2.4-2.7% gap) in shorter runtime. Within the DRL algorithm, DRL(10240) yields the lowest objective values at a higher runtime. Meanwhile, DRL(1024) achieves an optimality gap of less than $\sim$2% but is approximately ten times faster than DRL(10240). In the Rayleigh distribution scenarios, the objective value increases compared to Gaussian distribution across all methods, and the optimality gap between heuristic methods and RL policies further widens. While DRL(10240) delivers the best solutions, DRL(1024) offers highly desirable results with a less than $\sim$3% optimality gap in a faster solution runtime. In all instances, the greedy decoding approach underperforms, although it takes only a fraction of the runtime compared to other methods. In summary, despite the addition of extended road networks and an increased number of mission points, DRL produces better results (with the exception of DRL(greedy)) in significantly faster computation time (2-20 times faster than heuristics, depending on the decoding strategy) while solving larger neighboring problem sizes. Therefore, it can be concluded that the DRL algorithm demonstrates effective generalization across unknown scenarios.

D. Case study evaluation

As a case study, we apply our trained model in a simple simulation over a real-world task site consisting of 45 mission points located along a road network. Unlike the training instances, all the mission points in this scenario are situated on the road network, making them accessible to both the UAV and UGV and serving as potential rendezvous locations for UAV-UGV recharging. Given that DRL(10240) has demonstrated the best performance in terms of solution quality among DRL strategies, and GLS has emerged as the top performer among heuristic methods in previous analyses, we select these two methods to perform routing in this scenario. Fig. 4 illustrates the cooperative routes between UAV and UGV for the given scenario as obtained from the two methods. Recharging plays a crucial role in

optimizing coordination between UAV and UGV to achieve an optimal cooperative route. Hence, in Table III, we analyze the cooperative routes from both methods to understand their underlying UAV-UGV coordination aspects and recharge planning.

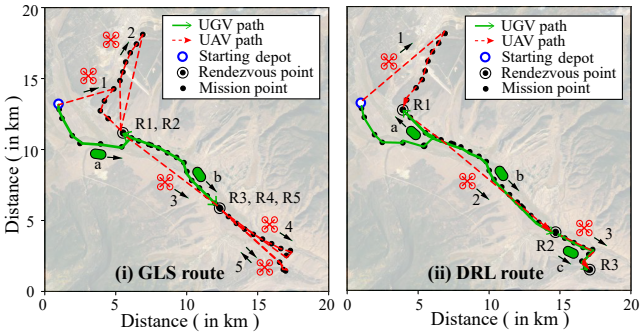

Fig. 4: Cooperative Routes of UAV and UGV in the case study scenario. The numerical and alphabetical sequences indicate the movement directions of the UAV and UGV, respectively. R1, R2, ..., RN denote the recharging instances between the UAV and UGV. The route animation can be found at `http://tiny.cc/hmnjxz`.

TABLE III: Comparison of cooperative routes in the case study scenario

Metrics	DRL(10240)	GLS
Mission time (min.)	148	200
No. of recharge stops	3	5
Recharge time (min.)	45	75
Waiting time (min.)	49.7	30.5
Idle time (min.)	94.7	105.5

It can be observed that the DRL policy optimizes the rendezvous between the UAV and UGV, requiring the UAV to recharge from the UGV only 3 times to complete the mission. In contrast, the GLS method results in 5 recharging instances. Frequent recharging forces the UAV to undertake frequent detours and endure longer recharging service periods, consequently extending the mission duration. Under the DRL policy, the UAV waits for an extended period (waiting time = 49.7 minutes) to minimize the need for frequent recharging and its associated longer service times (recharge service time = 45 minutes). Conversely, with the GLS method, UAV spends less waiting time of 30.5 minutes for the UGV, due to its frequent recharging. However, this approach results in longer service periods (75 minutes) and, ultimately, a longer mission period. The *idle time*, which represents the total waiting and recharging time, is 10.8 minutes longer with the GLS method. In addition to recharging, the routing pattern also contributes to the mission duration in this scenario. The suboptimal performance of the heuristic-based method can be attributed to its multi-level nature ('UGV first, UAV second'), as optimizing at an individual level does not necessarily result in an overall optimized route.

E. Dynamic Planning

We can leverage the trained DRL policy to address dynamic changes in the scenario by implementing it for online planning. In our case study, we explore the potential for dynamic planning when new mission points appear at random places during the routing process. These randomly appearing mission points are assumed to appear outside the road network and, hence, can only be visited by the UAV. A key assumption is that the UAV and UGV can share information only during their rendezvous process, simulating an operational and communication constraint. Therefore, after initial planning to establish the primary route, the trained policy updates its encoder space during each rendezvous, to include the new mission point and adjusts its actions in response to it. We opt for the DRL(greedy) method due to its rapid execution time, which makes it ideal for online planning. Fig. 5 illustrates the time-visuals of the UAV-UGV route from the DRL policy as they adapt their initial plan to incorporate the new mission points.

V. Conclusion & Future Work

In this study, we utilize a deep reinforcement learning-based planning framework that incorporates a transformer network with attention layers to address a fuel-constrained UAV-UGV cooperative routing problem. The problem involves both the UAV and UGV visiting a set of predefined mission points, with the UAV being periodically recharged by the UGV. Within the proposed transformer network, the

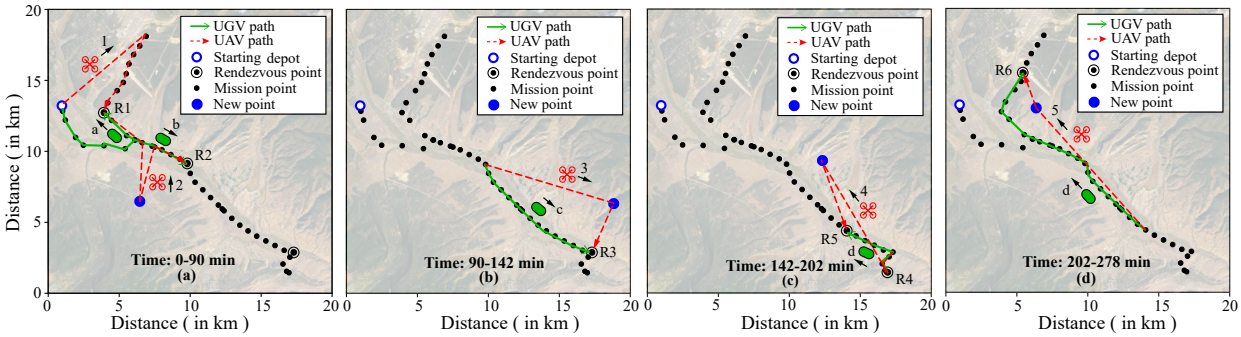

Fig. 5: Dynamic route planning for randomly appearing mission points. (a) Route replanned after the first recharging instance to visit the first random point. (b) UAV visits the second random point after the second recharging instance. (c) UAV visits the third random point after the fourth recharging instance. (d) UAV visits the fourth random point to conclude its mission. The route animation can be found at `http://tiny.cc/hmnjxz`.

encoder generates input embeddings from the input data, while the decoder determines actions for visiting mission points by leveraging these embeddings and the contextual state. The decoder also adopts a sortie-wise, one-agent-per-decoding strategy to accommodate the multi-agent aspect of the problem. Upon evaluation of test instances, our proposed framework: 1) Outperforms conventional multi-staged heuristic-based methods and existing learning baseline (AM) in solution quality for the tested problem instances. 2) Constructs UAV-UGV cooperative routes in a shorter runtime compared to the baseline methods. 3) Demonstrates its generalizability by producing better solutions in unknown scenarios of varying sizes and distributions. 4) Explores the potential of implementation as online planning to accommodate dynamic changes, as tested in a case study. In the future, we plan to extend the framework to solve long-duration, persistent mission planning and evaluate it using a real, physics-based simulator to account for stochasticities. We also aim to include multi-UAV-UGV scenarios, along with other learning-based baselines, for broader comparison and comprehensive analysis.

REFERENCES

[1] Yao Liu, Zhihao Luo, Zhong Liu, Jianmai Shi, and Guangquan Cheng. Cooperative routing problem for ground vehicle and unmanned aerial vehicle: The application on intelligence, surveillance, and reconnaissance missions. *IEEE Access*, 7:63504–63518, 2019.

[2] Daniel H Stolfi, Matthias R Brust, Grégoire Danoy, and Pascal Bouvry. Uav-ugv-umv multi-swarms for cooperative surveillance. *Frontiers in Robotics and AI*, 8:616950, 2021.

[3] Anuj Puri, KP Valavanis, and M Kontitsis. Statistical profile generation for traffic monitoring using real-time uav based video data. In *2007 Mediterranean Conference on Control & Automation*, pages 1–6. IEEE, 2007.

[4] Omer Ozkan and Muhammed Kaya. Uav routing with genetic algorithm based matheuristic for border security missions. *An International Journal of Optimization and Control: Theories & Applications (IJOCTA)*, 11(2):128–138, 2021.

[5] Md Safwan Mondal, Subramanian Ramasamy, James D Humann, James M Dotterweich, Jean-Paul F Reddinger, Marshal A Childers, and Pranav Bhounsule. A robust uav-ugv collaborative framework for persistent surveillance in disaster management applications. In *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1239–1246. IEEE, 2024.

[6] Chi Yuan, Youmin Zhang, and Zhixiang Liu. A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques. *Canadian journal of forest research*, 45(7):783–792, 2015.

[7] Mingjia Zhang, Huawei Liang, and PengFei Zhou. Cooperative route planning for fuel-constrained ugv-uav exploration. In *2022 IEEE International Conference on Unmanned Systems (ICUS)*, pages 1047–1052. IEEE, 2022.

[8] Jianqiang Li, Genqiang Deng, Chengwen Luo, Qiuzhen Lin, Qiao Yan, and Zhong Ming. A hybrid path planning method in unmanned air/ground vehicle (uav/ugv) cooperative systems. *IEEE Transactions on Vehicular Technology*, 65(12):9585–9596, 2016.

[9] Satyanarayana G Manyam, Kaarthik Sundar, and David W Casbeer. Cooperative routing for an air–ground vehicle team—exact algorithm, transformation method, and heuristics. *IEEE Transactions on Automation Science and Engineering*, 17(1):537–547, 2019.

[10] Subramanian Ramasamy, Jean-Paul F Reddinger, James M Dotterweich, Marshal A Childers, and Pranav A Bhounsule. Coordinated route planning of multiple fuel-constrained unmanned aerial systems with recharging on an unmanned ground vehicle for mission coverage. *Journal of Intelligent & Robotic Systems*, 106(1):30, 2022.

[11] Zheng Wang and Jiuh-Biing Sheu. Vehicle routing problem with drones. *Transportation research part B: methodological*, 122:350–364, 2019.

[12] Ziye Tang, Willem-Jan van Hoeve, and Paul Shaw. A study on the traveling salesman problem with a drone. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 16th International Conference, CPAIOR 2019, Thessaloniki, Greece, June 4–7, 2019, Proceedings 16*, pages 557–564. Springer, 2019.

[13] Kaarthik Sundar, Saravanan Venkatachalam, and Sivakumar Rathinam. Formulations and algorithms for the multiple depot, fuel-constrained, multiple vehicle routing problem. In *2016 American Control Conference (ACC)*, pages 6489–6494. IEEE, 2016.

[14] Yao Liu, Zhong Liu, Jianmai Shi, Guohua Wu, and Witold Pedrycz. Two-echelon routing problem for parcel delivery by cooperated truck and drone. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(12):7450–7465, 2020.

[15] Diego Cattaruzza, Nabil Absi, and Dominique Feillet. Vehicle routing problems with multiple trips. *4or*, 14:223–259, 2016.

[16] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[17] Jingwen Li, Yining Ma, Ruize Gao, Zhiguang Cao, Andrew Lim, Wen Song, and Jie Zhang. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. *IEEE Transactions on Cybernetics*, 52(12):13572–13585, 2021.

[18] Guohua Wu, Mingfeng Fan, Jianmai Shi, and Yanghe Feng. Reinforcement learning based truck-and-drone coordinated delivery. *IEEE Transactions on Artificial Intelligence*, 2021.

[19] Mingfeng Fan, Yaoxin Wu, Tianjun Liao, Zhiguang Cao, Hongliang Guo, Guillaume Sartoretti, and Guohua Wu. Deep reinforcement learning for uav routing in the presence of multiple charging stations. *IEEE Transactions on Vehicular Technology*, 2022.

[20] Subramanian Ramasamy, Md Safwan Mondal, Jean-Paul F Reddinger, James M Dotterweich, James D Humann, Marshal A Childers, and Pranav A Bhounsule. Heterogenous vehicle routing: comparing parameter tuning using genetic algorithm and bayesian optimization. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 104–113. IEEE, 2022.

[21] Prashant Sankaran, Katie McConky, Moises Sudit, and Hector Ortiz-Pena. Gamma: graph attention model for multiple agents to solve team orienteering problem with multiple depots. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[23] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[24] Arnon M Hurwitz, James M Dotterweich, and Trevor A Rocks. Mobile robot battery life estimation: battery energy use of an unmanned ground vehicle. In *Energy Harvesting and Storage: Materials, Devices, and Applications XI*, volume 11722, pages 24–40. SPIE, 2021.

[25] Parikshit Maini and PB Sujit. On cooperation between a fuel constrained uav and a refueling ugv for large scale mapping applications. In *2015 international conference on unmanned aircraft systems (ICUAS)*, pages 1370–1377. IEEE, 2015.

[26] Fernando Ropero, Pablo Muñoz, and María D R-Moreno. Terra: A path planning algorithm for cooperative ugv–uav exploration. *Engineering Applications of Artificial Intelligence*, 78:260–272, 2019.

[27] Parikshit Maini, Kaarthik Sundar, Mandeep Singh, Sivakumar Rathinam, and PB Sujit. Cooperative aerial–ground vehicle route planning with fuel constraints for coverage applications. *IEEE Transactions on Aerospace and Electronic Systems*, 55(6):3016–3028, 2019.

[28] Yu Wu, Shaobo Wu, and Xinting Hu. Cooperative path planning of uavs & ugvs for a persistent surveillance task in urban environments. *IEEE Internet of Things Journal*, 8(6):4906–4919, 2020.

[29] Google. Google OR-tools. `https://developers.google.com/optimization`, 2021. Online; accessed Feb 2, 2021.

[30] Md Safwan Mondal, Subramanian Ramasamy, and Pranav Bhounsule. A bilevel optimization framework for fuel-constrained uav-ugv cooperative routing: Planning and experimental validation. *arXiv preprint arXiv:2303.02315*, 2023.

[31] Md Safwan Mondal, Subramanian Ramasamy, James D. Humann, Jean-Paul F. Reddinger, James M. Dotterweich, Marshal A. Childers, and Pranav A. Bhounsule. Cooperative multi-agent planning framework for fuel constrained uav-ugv routing problem, 2023.