Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# RL-Assisted A-Teams for Adaptive Algorithm Selection in UGV-UAV Route Optimization

**Subramanian Ramasamy**

subramanian2698@gmail.com

University of Illinois Chicago

**Md Safwan Mondal**

University of Illinois Chicago

**James D. Humann**

DEVCOM Army Research Laboratory

**James M. Dotterweich**

DEVCOM Army Research Laboratory

**Pranav Bhounsule**

University of Illinois Chicago

**Research Article**

**Keywords:** Multi-agent planning, Reinforcement Learning, Vehicle Routing Problem, Hyper-Heuristics, UAV, UGV

**Posted Date:** September 11th, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-10482875/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** No competing interests reported.

# RL-Assisted A-Teams for Adaptive Algorithm Selection in UGV-UAV Route Optimization

Subramanian Ramasamy[1*], Md Safwan Mondal[1], James D. Humann[3], James M. Dotterweich[2], Pranav A. Bhounsule[1]

[1*]Department of Mechanical and Industrial Engineering, University of Illinois Chicago, Chicago, 60607, IL, USA.
[2]DEVCOM Army Research Laboratory, Aberdeen Proving Grounds, Aberdeen, 21005, MD, USA.
[3]DEVCOM Army Research Laboratory, Minneapolis, MN, USA.

*Corresponding author(s). E-mail(s): subramanian2698@gmail.com;
Contributing authors: mdsafwanmondal@gmail.com; james.d.humann.civ@army.mil; james.m.dotterweich.civ@army.mil; pranav@uic.edu;

**Abstract**

Unmanned Aerial Vehicles (UAVs) are capable of monitoring large areas but are constrained by limited battery life. Collaborating with Unmanned Ground Vehicles (UGVs) can greatly extend the endurance of UAVs by utilizing UGVs as mobile charging platforms. Integrating Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) enables effective long-duration surveillance over large areas, especially where real-time adaptability is essential. However, this also increases the complexity of routing these heterogeneous vehicles. Therefore, developing an efficient UGV-UAV routing framework is essential to ensure real-time adaptability, enabling them to respond effectively to dynamic changes. In this paper, we introduce a novel learning-based hyper-heuristic approach using Reinforcement Learning (RL) within a multi-agent optimization framework (A-Teams) to deliver computationally efficient, real-time UGV-UAV routing solutions. Hyper-heuristics are high-level optimization methods that select heuristic or meta-heuristic algorithms to solve complex computational problems more

efficiently across different problem domains. The RL-assisted optimization framework is validated through various scenarios and tested for generalizability across different numbers of UAVs by comparing its performance with three alternative methods: (1) Genetic Algorithm, (2) conventional A-Teams, and (3) A-Teams variant. The proposed framework produces near-optimal solutions that are computed around 30-70% faster than the other methods while maintaining the solution quality and providing better solutions in some cases than its counterparts. Finally, the proposed method is evaluated on a real-world civilian application through a case study involving collaborative UAV-UGV routing for bridge inspection under dynamically changing conditions. Results demonstrate that the method efficiently adapts to changes, generating solutions with re-optimization times as low as 3 minutes for scenarios whose planning horizon is over 90 minutes for 50 task points.

**Keywords:** Multi-agent planning, Reinforcement Learning, Vehicle Routing Problem, Hyper-Heuristics, UAV, UGV

# 1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have rapidly gained popularity across fields such as entertainment, logistics, surveillance, and disaster relief, thanks to their compact size, relatively low cost, agility, and ease of use [1]. Despite these advantages, a primary limitation of UAVs is their restricted operational time due to limited battery capacity, which constrains their ability to perform critical tasks, particularly those requiring extensive coverage or prolonged duration, like surveillance and disaster relief.

To extend operational endurance over larger areas, UAVs can be paired with Unmanned Ground Vehicles (UGVs), allowing UGVs to serve as mobile recharging stations [2]. In this cooperative setup, UAVs can leverage their speed and altitude to provide a comprehensive aerial view, returning to UGVs for recharging. The collaborative routing of UAV-UGV systems, however, presents an NP-Hard combinatorial optimization challenge [3]. Traditional route planning algorithms often struggle with real-time planning due to the inherent complexity of combinatorial optimization. To address this, we propose an RL-policy-based hyperheuristic method integrated into an existing optimization framework, where the RL policy functions as a high-level decision maker on algorithm selection, which are hyper-heuristic parameters, enabling end-to-end autonomous planning that is faster and more efficient for UAV-UGV collaborative routing.

# 2 RELATED WORK

Several works in the literature have addressed the cooperative UGV-UAV routing problem. Liu et al. [4] addressed a cooperative Ground Vehicle (GV)-UAV routing problem for surveillance and reconnaissance missions. Their approach initially generates an extensive UAV route without considering battery constraints. A Split Heuristic is then applied to divide this route into feasible segments, which are linked by GVs

with necessary constraints to complete the mission. Gao et al. [5] formulated a cooperative UGV-UAV routing problem for emergency resource delivery during COVID-19. Their system processes operation orders through an intelligent module, generate routes using a Mixed Integer Linear Programming (MILP) approach, and refines these routes through an iterative improvement algorithm. Seyedi et. al. [6] addressed the persistent surveillance problem using energy-constrained UAVs supported by UGVs, acting as mobile recharging stations. The UAVs and UGVs were organized into separate UGV-UAV teams, with the environment optimally partitioned into segments for each team to perform persistent cyclic surveillance of task points. Within each segment, the UGV-UAV team optimally visited the task points, utilizing the Dantzig-Fulkerson-Johnson (DFJ) linear programming algorithm. Consequently, these studies are limited in using standalone optimization algorithms for UGV and UAV routing, missing out on the potential benefits of combining different algorithms. The combination can improve the solution and the computational efficiency by taking advantage of the strengths of each algorithm together [7].

Some works in the literature have also investigated re-planning UAV-UGV paths owing to dynamic changes in the environment. Ni et al. [8] introduced an online path planning method for mixed UAV-UGV systems using a Dragonfly algorithm inspired by nature to optimize movements in 3D space, and tested in 3D simulations. Ma et al. [9] developed a fast re-planning method for obtaining optimal UAV paths between the start and goal. The work presented an improved A* algorithm to perform real-time path re-planning. Martinez et. al. [10] tackled a real-time navigation problem for urban firefighting with UAVs and UGVs in GNSS-denied areas. They used Monte-Carlo localization, Lazy Theta* algorithm, and 3D-LIDAR mapping to enable rapid navigating planner adjustment based on new obstacles or changes, ensuring efficient navigation. Both UGV and UAV uses the same set of algorithms, but UGV planning is done in 2D whereas the UAV planning is done in 3D.

Beside standalone optimization approaches, there are multi-agent and hybrid optimization algorithm frameworks studied for combinatorial optimization problems, including Vehicle Routing Problem (VRP). A-Teams is a multi-algorithm framework that uses a team of optimization agents such as Constructor, Improver, and Destroyer agents to evolve a solution pool towards near-optimal results. In this context,"agents" refer to the role of algorithms in the framework. Jedrzejowicz et al. [11] implemented this framework to solve a Resource Investment Problem in which the Improver agents use different optimization algorithms like Local search, Lagrangian relaxation, Path relinking algorithms, Crossover operators and cooperate together to solve such a problem. Other multi-agent optimization frameworks are used across the literature to solve combinatorial optimization problems. For instance, Milano and Roli et al. [12] proposed the MultiAGent Metaheuristic Architecture (MAGMA), an optimization framework that integrates Memetic algorithms and GRASP (Greedy Randomized Adaptive Search Procedures). In MAGMA, agents were operated at various levels: Solution Builder Agents applied constructive heuristic algorithms, Solution Enhancer Agents carried out local searches, Strategy Agents devised strategies to avoid local optima, and Coordination Agents oversaw the search process and agent coordination. In previous work, Ramasamy et al. [13] introduced a variant of the A-Teams framework

by incorporating a new "Predictor Agent" that uses a supervised machine learning model to predict the feasibility of a solution before it gets evaluated for its fitness.

There are some learning based approaches utilized in other multi-agent optimization frameworks. In Lotfi and Acan [14], Learning-Based Multi-Agent System (LBMAS) for solving combinatorial optimization problems is presented. The LBMAS architecture includes two different types of Agents: Metaheuristic Agents and System Agents. The Metaheuristic Agents are responsible for applying their own search strategies and providing discovered solutions. There are seven metaheuristics implemented by the Metaheuristic Agents. There are four system agents, each responsible for specific tasks within the multi-agent system. For example, in System Agents, the Manager Agent is responsible for selecting the metaheuristic to be used in the next iteration, based on the Russian roulette selection principle, and evaluating improvement levels of metaheuristics in the objective function.

In multi-agent systems with multiple algorithms, not all algorithms perform well across all iterations due to varying problem-specific characteristics. This variability led researchers to develop a higher-level approach for managing algorithm selection throughout the optimization process, resulting in the concept of "hyperheuristics," first introduced by Cowling et al. [15]. Some research has been done in exploring the capabilities of hyperheuristics for solving various combinatorial optimization problems. Cowling et al. [16] investigates a genetic algorithm-based hyperheuristic (hyper-GA) for scheduling geographically distributed training staff and courses. The aim of the hyper-GA is to evolve a good-quality heuristic for each given instance of the problem and use this to find a solution by applying a suitable ordering from a set of low-level heuristics. The optimization problem addressed in this paper is the trainer scheduling problem involving the allocation of staff to timeslots and locations. Grobler et al. [17] explores the concept of heuristic space diversity within a meta-heuristic algorithm in pursuit of improved performance benefits. The study evaluates various strategies for management of heuristic space diversity, highlighting the impact it has on hyper-heuristic performance.

In recent years, learning-based methodologies for hyper-heuristics is being implemented for solving VRP problems as such feature can help in the search process, giving the optimization agents more autonomy in taking decisions. One of the earliest works in implementing learning-based hyper-heuristics is from Sabar et al. [18] introduces a new hyper-heuristic framework called Dynamic Multiarmed Bandit-Gene Expression Programming Hyper-Heuristic, which is a high-level strategy that uses a dynamic multiarmed bandit for Low-level heuristic selection. The LLH comprises of simple heuristics like swapping the nodes in different configurations such as 2-opt, 3-opt etc. This framework is applied to the Dynamic VRP (DVRP) and achieved competitive results.

With the recent advancements in deep learning approaches, researchers have implemented the Reinforcement Learning (RL) agents for performing heuristic selection in an optimization process and obtained promising results. Early works deal with performing algorithm selection using RL methods for other types of computer science problems. Lagoudakis et al. [19] explores the use of Deep Q-Learning to dynamically select sorting algorithms, such as Bubble Sort, Merge Sort, Shell Sort, and Quick Sort,

during the sorting process. By applying reinforcement learning to adaptively choose a better sorting algorithm at each step, the approach improves computational efficiency compared to using a single fixed algorithm. This adaptive decision-making leverages the strengths of each sorting method based on the size of a list that's sorted, resulting in enhanced sorting performance. In Armstrong et al. [20], RL is applied to computational chemistry simulations of gaseous reactions. A policy gradient agent with epsilon-greedy selection adaptively chooses between two algorithms, $O(n^2)$ and $O(n)$, based on the simulation's current state. This dynamic selection enhances real-time computational efficiency.

There are also some research work that has been done in implementing RL as hyper-heuristic agent for heuristic selection. Garrido et al. [21] utilizes a self-adaptive hyper-heuristic to solve static and dynamic instances of the capacitated vehicle routing problem (CVRP). The hyper-heuristic manages a generic sequence of constructive and perturbative low-level heuristics (LLHs), such as the 2, 3-opt operator, gradually applied to construct or improve partial routes. This approach has been tested using standard benchmarks and compared with previous hyper-heuristics and well-known methods proposed in the literature. It has shown the ability to guide the search for appropriate operators and adapt to dynamic scenarios more naturally than other methods.

Yao et al. [22] proposes a multi-objective hyper-heuristic (MOHH) framework for walking route planning in a smart city. The search framework features a set of LLHs to generate new routes, along with a reinforcement learning mechanism to select good low-level heuristics and accelerate the search speed. The low-level heuristic deals with different modes of hop transitions such as 1, 2, 3-hop path sets. This work addresses the multi-objective route planning problem, where the goal is to provide a safe walking route in addition to distance and time considerations.

Mosedegh et al. [23] introduce a novel Hyper Simulated Annealing (HSA) for the mixed-model sequencing problem with stochastic processing times in a multi-station assembly line. The HSA employs a Q-learning algorithm to select appropriate heuristics through its search process. The heuristics used are swapping (SW), shifting (SH), knowledge sharing (KS), and a random selection (RA) heuristic. The optimization problem involves minimizing the weighted sum of expected total work-overload and idleness in the assembly line. The results proved the competitiveness of HSA method compared to other methods and performs superior over the simulated annealing algorithms.

The novelty of this work is to propose a RL-Assisted A-Teams (RAAT) framework, which utilizes Reinforcement Learning as a hyper-heuristic to select one or multiple "metaheuristic" or optimization algorithms at the lower level to address a collaborative UAV-UGV heterogeneous routing problem. While the above methods in the literature demonstrate promising results in applying RL within hyper-heuristic frameworks, they primarily uses simpler heuristics as low-level algorithms and is limited to basic instances of Vehicle Routing Problems. To the best of our knowledge, incorporating a learning-based model as an agent within a multi-agent optimization framework was a novel approach that had not been explored. The proposed method is

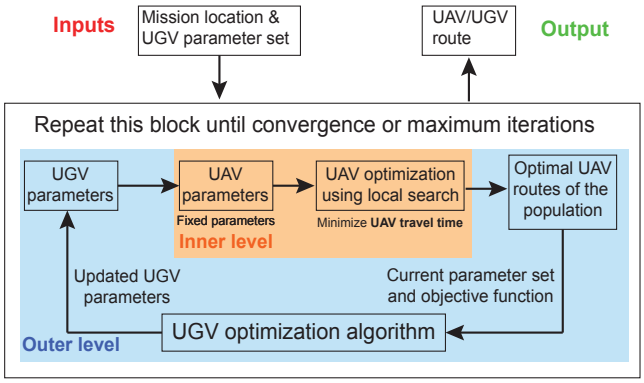



**Fig. 1:** Overview of the bi-level optimization algorithm.

evaluated against various approaches and tested on diverse problem sizes and distributions, demonstrating its generalizability. To this end, we present the following novel contibutions:

1. The RL-agent chooses one or a set of different optimization algorithms in the multi-agent framework, thereby acting as a high-level decision maker. This imparts autonomy to the framework.
2. The RL-agent strategically chooses which optimization algorithm to be used in each step of the optimization, thus leading to an efficient selection of optimization algorithms. This imparts computational efficiency to the method, enabling the framework to adapt to dynamic changes.
3. This approach is evaluated through a bridge inspection case study simulation using a collaborative UAV-UGV system, addressing a practical civil engineering problem with a heterogeneous autonomous vehicle team.

# 3 METHODS

## 3.1 Problem statement

To address the UGV-UAV coordinated routing problem for surveillance applications, a set of task points is considered, $\mathcal{M} = \{m_0, m_1, \ldots, m_n\}$, to be visited both by UAV and UGV. This UGV-UAV system consists of a set of UAVs $\mathcal{A} = \{u_i^a, \forall i = 1, ..., K\}$ and one UGV persistently visiting the task points $\mathcal{M}$. The UGV and UAV start from the depot $D$. For practical application purposes, these points are considered for persistent surveillance by both UAV and UGV. The UAV can either get recharged at $D$ or by making a rendezvous with the UGV. The vehicle system is of a heterogeneous nature where UAVs have a higher velocity $v_a$ but is limited in its energy capacity $E_a$, whereas the UGV has a slower velocity $v_g$, while having a larger energy capacity $E_g$ to support UAV recharging and power its components. This approach is taken in this study in order to extend the range of UAVs by teaming up with UGV. The operational range of the UAV with full energy $E_a$ is visualized as blue circles in Figure 2 (a), showing

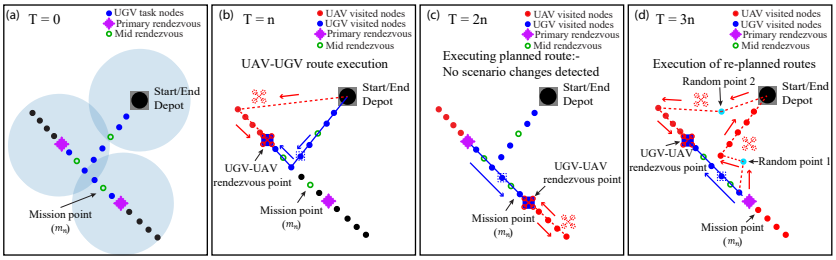



**Fig. 2:** Persistent surveillance illustration for a collaborative UGV-UAV routing problem. For a single Persistent Surveillance mission, steps (a) through d) occur sequentially, where 'n' represents the timestamp of the route. a) A candidate UGV-UAV task visit points. The blue circles in (a) represents radial coverage that the UAV can make in a single charge. (b), (c), (d) Execution of the planned/re-planned route. Steps (c) and (d) are repeated for persistent node visits. Re-planning is done online if dynamic changes are observed.

that a single UAV cannot cover all task points without recharging. This observation highlights the need for strategic UGV routing to facilitate timely rendezvous and ensure complete task coverage. Consequently, the strong interdependence between the UAV and UGV routes naturally leads to a bi-level optimization problem, where the UGV route is optimized at the outer level and the corresponding UAV route is optimized at the inner level following a "UGV first, UAV second" strategy. Figure 1 represents the bi-level optimization methodology as an overview.

The UGV route optimization is formulated in terms of a set of free parameters that represent the rendezvous locations between the UAV and the UGV. The complete parameter set for a candidate UGV route, $X_s$, is made up of two subsets: primary rendezvous points $X_p = \{x_1, x_2, \ldots, x_j\}$, which facilitate UAV recharging and directing towards different task sub-regions, and mid-rendezvous points $X_m = \{x_1, x_2, \ldots, x_k\}$, intended for UAV recharging along the way of UGV's traversal. This would help minimize the waiting time that the UGV makes along the road network. Primary rendezvous points are identified by solving the Minimum Set Cover (MSC) Problem through Constraint Programming [24]. This approach effectively addresses the NP-Hard nature of the MSC problem, allowing for multiple $X_p$ sets depending on the scale and complexity of the scenario. Equations 3.1-3.3 represents the condition required to obtain a single $X_p$ set such that each location in $X_p$ should have radial coverage $R$ whose union encompasses all the mission points $\mathcal{M}$. $R$ is obtained from half the UAV's flight time $T_f^a$ and velocity $v_a$ of the UAV. Mid-rendezvous points ($X_m$), which support UAV recharging during transit, are randomly selected between each location in $X_p$. The total number of route parameters is represented as $\mathcal{S} = |X_s| = j + k$. For each candidate UGV route evaluated at the outer level, the corresponding UAV route is computed by solving an Energy-Constrained Vehicle Routing Problem (E-VRP) subject to the prescribed UGV rendezvous locations and UAV energy limitations. The resulting UAV solution is subsequently fed back to the outer-level optimizer, where its feasibility and objective value are used to evaluate the candidate UGV route. A solution is deemed feasible only if the UAV visits every task point at least once while satisfying all energy constraints.

$$\min \quad j \tag{3.1}$$

s.t.,

$$\bigcup_{i=1}^{j} \{x_i \mid |x_i - m_n| \leq R,\, x_i \in X_p\}, \forall m_n \in \mathcal{M} \backslash x_i \tag{3.2}$$

where

$$j = |X_p|, k = |X_m| = j + 1, R = 0.5 \cdot T_f^a \cdot v_a \tag{3.3}$$

The 0.5 in 3.3 represents the radial coverage distance from a task point $m_n$ to the rendezvous point $x_i$. To do the persistent surveillance, the task points must be routinely visited by either the UAV or UGV. This constitutes multiple visits of each task point, with the age period $a^d = t - t^d_{\text{last}}$ determining the elapsed time gap between two consecutive visits of node $m_d$ in minutes. To avoid a task point not being visited for a while, we keep track of a *score metric* $S$ as:

$$S = \frac{1}{3600} \sum_{d=1}^{n} \sum_{q=1}^{l} \left(t_q^d - t_{q-1}^d\right)^2 \tag{3.4}$$

Here $t_0^d, t_1^d, ..., t_l^d$ are the time instances in ascending order when a task node $m_d$ is visited multiple times. The $t_0^d$ is the time at the beginning of the mission and $t_l^d$ is the latest time that the node $m_d$ was visited during the end of the mission $T$. The $S$ is quadratic in time, thus penalizing nodes with longer visit intervals. Hence, the **objective** of this persistent surveillance problem is to minimize $S$ so that all the nodes are visited frequently with minimal intervals between successive visits.

Figure 2 (a) through (d) shows the UGV-UAV persistent surveillance nature for a toy scenario. For Figure a), the initial route planning would happen by solving the optimization problem, and the planned route gets executed as shown in Figure b). When the UGV and UAV are at an $X_p$ location, the scenario is assessed for any dynamically changed or newly appeared task points, represented as $\mathcal{R} = \{r_0, r_1, \ldots, r_n\}$. If there's no dynamic change, the route gets executed per the initial plan as illustrated in Figure c). If such changes are identified, the optimizer performs re-planning, and the UGV and UAV then execute the updated route sequence as illustrated in Figure d). This entire process happens persistently until the mission end time criteria $T$ is met.

This study evaluates four different approaches for optimizing the outer-level UGV route, while the inner-level UAV routing problem is consistently formulated as an Energy-Constrained Vehicle Routing Problem (E-VRP) and solved using OR-Tools. The four methods differ solely in their approach to UGV route optimization, one of which is the proposed method introduced in this work. Since the UAV route is inherently dependent on the selected UGV route, improving the efficiency of the outer-level UGV optimization reduces the number of computationally expensive UAV route evaluations, thereby lowering the overall computational cost without compromising solution quality. The following sections first describe the inner-level UAV optimization process, followed by the four UGV optimization methods.

### 3.2 Solving E-VRP for inner-level UAV routing

The inner-level UAV route optimization is defined as an Energy-Constrained Vehicle Routing Problem (E-VRP) to obtain a near-optimal path for the UAV subjected to satisfying its energy and time window constraints. The MILP formulation is as follows. Consider a directed graph $G = (V, E)$ where $V$ is the set of UAV task points $V = \{0, 1, 2, ...., m, D\}$ and $E$ is the set of edges that gives the arc costs between two consecutive nodes $i$ and $j$ and $E = \{(i, j)|i, j \in V, i \neq j\}$. Here, $D$ denotes the potential UGV points that the UAV could utilize to recharge. Let $c_{ij}$ be the non-negative arc cost between a particular $i$ and $j$. Let $t_{ij}$ be the travel time cost between a particular $i$ and $j$. Let $x_{ij}$ be the binary variable where the value of $x_{ij}$ will be 1 if a vehicle travels from $i$ to $j$, and 0 otherwise. The UAV routing problem is formulated as a VRP problem with fuel constraints, time windows, and dropped visits. The time windows constraint on the UAV comes from the UGV travel and rendezvous points, where the landing and takeoff time is restricted based upon the UGV traveling to a certain task point so that the UAV-UGV rendezvous could happen. The concised formulation for E-VRP is presented here, and more details may be found in [25].

Objective:

$$\min \sum_{i \in V} \sum_{j \in V} t_{ij} x_{ij} \tag{3.5}$$

s.t.,

$$f_j \leq f_i - (P^a t_{ij} x_{ij}) + L_1(1 - x_{ij}), \forall i \in V, j \in V \setminus D \tag{3.6}$$

$$f_j = Q, \quad \forall j \in D \tag{3.7}$$

$$0 \leq f_j \leq Q, \quad \forall j \in V \tag{3.8}$$

$$t_j \geq t_i + ((t_{ij} x_{ij})) - L_2(1 - x_{ij}), \forall i \in V, j \in V \tag{3.9}$$

$$t_j^l \leq t_j \leq t_j^u, \quad \forall j \in V \tag{3.10}$$

$$x_{ij} = 1 \rightarrow \sum_{i \in V \setminus D} x_{ji} = 1, \forall j \in D, \forall i \in V \setminus D \tag{3.11}$$

The objective is Eq. 3.5 is to minimize the time to complete UAV routing. The constraint in Eq. 3.6 is the Miller-Tucker-Zemlin (MTZ) formulation [26] for sub-tour elimination, which enables that none of the UAV batteries are fully drained out while eliminating subtours. $P^a$ in this equation represents the UAV power consumption curve, which will be provided in Section 4. Constraint Eq. 3.7 states that if the vertex is a recharging UGV stop, UGV must recharge the UAV to its full capacity $Q$. Constraint Eq. 3.8 is the condition that the UAV's charge at any vertex in $V$ should be between 0 and maximum capacity. Constraint Eq. 3.9 denotes that the cumulative arrival time at the $j^{th}$ node is equal to the sum of the cumulative time at the node $i$, $t_i$ and the travel time between nodes $i$ and $j$, $t_{ij}$. Constraint Eq. 3.10 is the time window constraint that restricts the vehicle to visit a certain vertex in the specified time window for that node. The constraint in Eq. 3.11 ensures that if any UAV comes to the refuel vertex to recharge, an arc must exist between that recharging node and

a task node to maintain the flow conservation. The above MILP formulation takes significant time to solve with the increased number of task points, which becomes unrealistic for practical hardware implementation. Ramasamy et al. [27] conducted a research by testing this MILP method on various toy scenarios, finding that solving a relatively simpler UGV-UAV routing problem involving 25 task points demanded considerable computational effort, with computation times being up to 30 times slower than the Constrained Programming (CP) approach while the optimality gap is up to 8% on average. Hence, CP with local search heuristics is used to solve this E-VRP problem quickly. Google's OR-Tools solver is utilized to perform the UAV optimization using this local search method. More details can be found in [27].

## 3.3 UGV outer-level routing: Comparitive approaches

### 3.3.1 Method 1: GA at outer-level and Constraint Programming at inner-level

Genetic algorithms (GAs) are meta-heuristic techniques inspired by natural selection of the fittest solutions, effectively solving global optimization problems that may contain multiple local optima. In this case, the outer-level block at the top implements GA to choose a UGV route. The UGV route heuristics is formulated as a set of free parameters illustrating candidate UGV routes. For each UGV route sent, the inner-level optimizes the UAV route using Constrained Programming (CP) method in Google's OR-Tools solver [28]. OR-Tools uses local search heuristics and solves using the CP approach. Some of the constraints, such as time constraints on the UAV task points, eventually come from the UGV route. UAV route optimization is closely linked to UGV route optimization, which utilizes genetic algorithms (GAs) based on the feasibility of the combined UGV-UAV route. A route is deemed feasible if it covers all the mission points considered for the persistent surveillance. The process iterates until the GA can no longer improve the solution or reaches the maximum iteration limit. The sample size of the population is considered to be $\mathcal{N}= 40$, which is chosen as per [29]. More details about this can be found in our previous work [25].

### 3.3.2 Method 2: Conventional A-Teams at outer-level and CP at inner-level

A-Teams is a multi-agent framework that uses a team of autonomous agents to optimize a given problem. Each agent in this framework possess distinct functionalities, solve combinatorial optimization problems through cooperation, and achieve solutions that are better than their individual counterparts. There are main core agents that constitute this framework: **Constructor Agent** is used to develop an initial pool of solutions using the construction algorithms or randomization. **Improver Agent** is used to improve the pool of solutions obtained from Constructor Agent using different optimization methods. **Destroyer Agent** is used to discard non-optimal, and redundant solutions. **Populations** are shared repositories for storing solutions that are computed and evaluated by different agents. Those solutions are also accessible to all the agents involved in the framework.

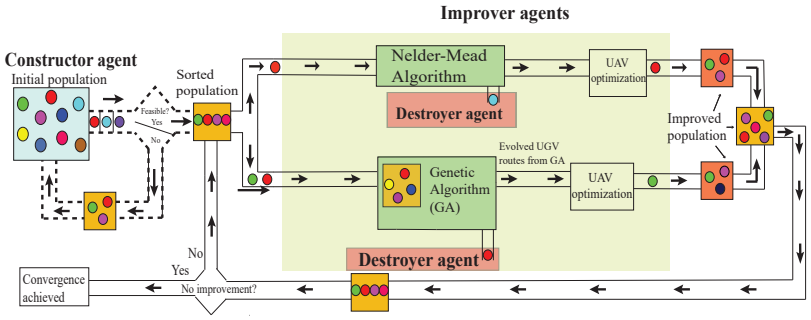



**Fig. 3:** Description of the conventional A-Teams optimization framework used for UGV-UAV routing. This A-Teams inspired from the works of Talukdar et. al [30] has Constructor, Improver, Destroyer and Predictor agents. Agents strategically choose UGV routes, which are fed into UAV optimization to get collaborative UGV-UAV route outputs.

Figure 3 shows the optimization using A-Teams for outer-level UGV routing. To solve this collaborative UGV-UAV routing problem, our previous work [31] had implemented this framework where the Constructor Agent creates an initial population by random initialization of candidate UGV routes using Latin Hypercube Sampling (LHS). The sample size of the Constructor agent is considered to be $\mathcal{N}= 40$ to be consistent with section 3.3.1. Those UGV route parameters are sent to the inner-level block and UAV optimization is solved using OR-Tools. The Improver Agents iteratively refine the UGV route based on the quality of the corresponding UAV routes until convergence, ultimately yielding an optimal collaborative routing solution. Here, Nelder-Mead and GA are used as Improver agents. For each tuned UGV route parameter given, the UAV optimization is carried in ther inner-level via Constrained Programming method using OR-Tools solver.

### 3.3.3 Method 3: A-Teams with Predictor Agent at outer-level and CP at inner-level

Figure 4 a) represents a variant of the A-Teams framework to perform UGV-UAV optimization. Unlike section 3.3.2, there are four agents in this variant: **Constructor Agent** is used to develop an initial pool of candidate UGV routes through randomization. **Improver Agent** is used to improve the pool of UGV routes obtained from Constructor Agent using different optimization methods. **Destroyer Agent** is used to discard non-optimal or infeasible or redundant UGV routes once it gets feedback from UAV optimization. **Predictor Agent** predicts the infeasible UGV routes before being evaluated by UAV optimization using Machine Learning algorithms. **Populations** are shared repositories for storing computed solutions and assessed by different agents. More details about this A-Teams variant can be found in [13].

To solve the collaborative UGV-UAV routing problem, the Constructor Agent initializes candidate UGV routes using Latin Hypercube Sampling (LHS) with a sample size of $\mathcal{N}= 40$. Similar to the previous methods, each candidate UGV route is forwarded to the inner-level optimization block, where the corresponding UAV route is

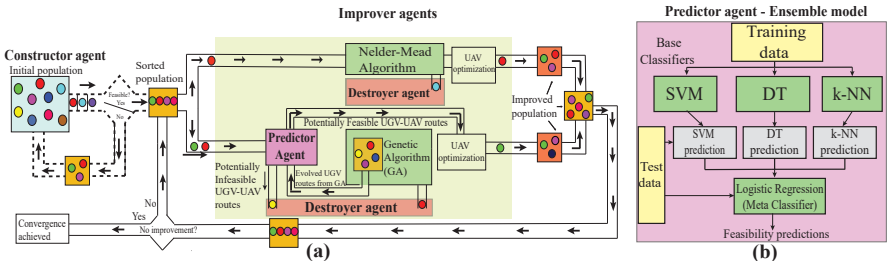



**Fig. 4:** Description of the proposed A-Teams optimization framework used for UGV-UAV routing. a) This proposed A-Teams has Constructor, Improver, Destroyer and Predictor agents. Agents strategically choose UGV routes, which are fed into UAV optimization to get collaborative UGV-UAV route outputs. b) Training of Ensemble model in Predictor Agent. Here SVM represents Support Vector Machine classifier, DT represents Decision Trees and k-NN represents k-Nearest Neighbors classifier.

optimized using OR-Tools. Feedback on route feasibility is used by Improver Agents, employing Nelder-Mead and Genetic Algorithms, to refine the UGV routes until near-optimal solutions are achieved. The novelty involves the inclusion of an additional Predictor agent in the framework. The primary role of the Predictor agent is to forecast the feasibility of combined UGV-UAV routes, reducing computational effort and improving efficiency. In the proposed system, the Predictor agent uses an Ensemble model of several base classifiers such as Support Vector Machines (SVM), Decision Trees (DT), and k-nearest Neighbors (k-NN) stacked together and uses Logistic Regression as the meta-classifier. Figure 4 b) shows the training process, where base classifiers are trained first, followed by the meta-classifier. The Constructor Agent randomly initializes UGV routes, which are then optimized for UAVs, creating a dataset of combined route solutions. This dataset, with a training batch size of $\mathcal{N}$ includes UGV route parameters as input features and binary feasibility labels 1/0 as output. Feasibility is determined by whether all task points are subjected to persistent surveillance. The Ensemble model, tested with UGV routes generated by a Genetic Algorithm, filters out infeasible routes before UAV optimization, thereby enhancing process efficiency and facilitating faster optimization.

### 3.3.4 Method 4: Reinforcement Learning framework for High-level Decision Making

This section outlines the proposed Deep Reinforcement Learning (DRL) implemented on A-Teams to solve the UAV-UGV cooperative persistent surveillance problem. So far in the current A-Teams optimization framework, there are several algorithms that performs the optimization concurrently at each to yield an optimal result. However, it is not always necessary to deploy every algorithm in every iteration of the optimization process. In certain steps of the optimization process, some algorithms of the currently implemented A-Teams, like Nelder-Mead or Genetic Algorithm, may add minimal improvement to the solution despite being used throughout, resulting in unnecessary function evaluations. In this work, we propose a DRL-based framework that functions as a high-level decision-making system, dynamically learning an optimal policy for

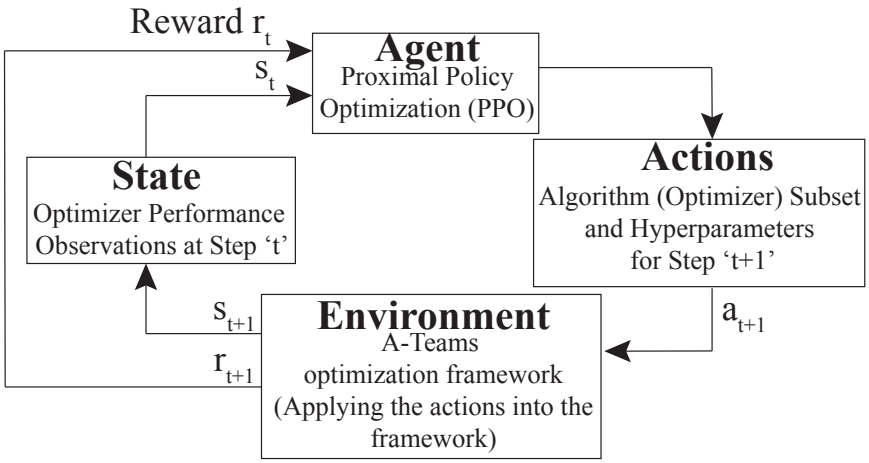



**Fig. 5:** Markov Decision Process in this study

algorithm selection at each stage of the optimization process. The decision in a step (action) corresponds to choosing an algorithm based on computational time, algorithm performance, and the objective function values from the previous step (state). A detailed explanation will be given in the following subsection. As mentioned in the previous subsections, the A-Teams optimization framework involves Improver agents containing Nelder-Mead algorithm (Local optimizer) and Genetic algorithm (Global optimizer), and Predictor agent containing supervised Ensemble Learning model. During each iteration of the optimization process, the DRL policy determines a specific action, represented by a selected combination of these algorithms, to apply in that step.

DRL is implemented as a sequential decision making problem where the RL agent sequentially selects the subset of algorithms to take part in each step of the optimization process. The system can be modeled as Markov Decision Process (MDP) where its components are defined by the tuple $< State, Actions, Reward, Policy >$. The flowchart representation of MDP is defined in Figure 5. The environment in which the DRL agent is implemented is the A-Teams framework.

**1) State space:** Since the environment for DRL is the optimization framework, the state space consists of observation information from the optimization process. The observations from the environment include the performance status of the algorithms used and the status of the UAV-UGV rendezvous locations in their routing. For this study, the state space is considered to have 13 components: **(i) Rendezvous points**, indicating the optimal UAV-UGV rendezvous locations for the current iteration's best solution obtained; **(ii) Local best**, representing the current best solution from the local optimizer; **(iii) Local best improved flag**, representing the binary variable that becomes 1 when the current local best solution is better than the best solution of the previous state, else 0. This observation provides an information to the policy that local optimizer at that state has a good contribution towards finding better solutions; **(iv) Global best**, representing the current best solution from the global

optimizer; **(v) Global best improved flag**, representing the binary variable that becomes 1 when the current global best solution is better than the best solution of previous state, else 0; **(vi) Current best**, representing the best solution of the current state; **(vii) Action step**, representing the action chosen for this time step to generate the current observations; **(viii) Population size**, representing the number of UAV-UGV optimizations performed; **(ix) Local optimizer size**, representing the number of UAV-UGV optimizations performed by local optimizer; **(x) Global optimizer size**, representing the number of UAV-UGV optimizations performed by global optimizer; **(xi) Predictor accuracy**, representing the accuracy of the ML model used for predictions; **(xii) Global best without Predictor agent**, representing the best solution of current state from global optimizer without using Ensemble learning model. This helps to gauge the quality of the Predictor Agent used in the optimization; **(xiii) Predictor agent flag**, representing the binary variable to see if the Predictor Agent is used in the current state or not.

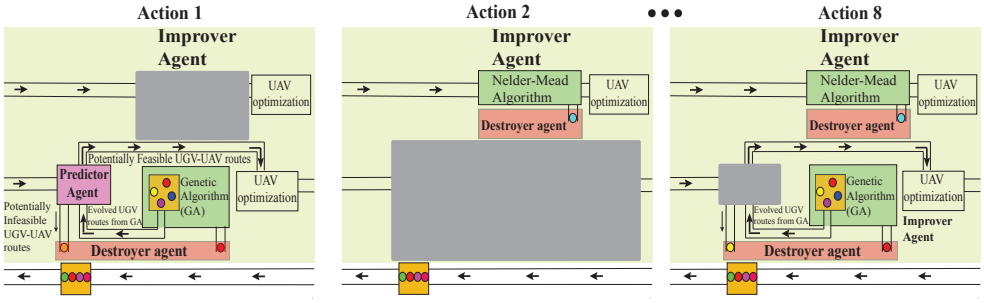



**Fig. 6:** Action space for DRL based approach. In the figure, Action 1 represents choosing the Global Optimizer and Predictor Agent to perform optimization; Action 2 represents choosing only the Local Optimizer to perform optimization; Action 3 represents choosing both the Local and Global Optimizer without choosing the Predictor Agent to perform optimization. The figure shows the Action 1, Action 2 and Action 8 out of the entire action space.

**2) Action space:** The action space is a discrete set of actions, each representing the algorithm or subset of algorithms to be applied in the next step of the optimization process. Alongside the algorithms to be selected for optimization, there are some hyperparameters that can be varied for the optimizers used in A-Teams. In this study, the hyperparameter maximum function evaluations 'Max' for the local optimizer Nelder-Mead algorithm is varied between two values: 5 and 10. Hence, the 8 discrete set of actions are as follows:

- Action 1: Use the combination of Global and local optimizers with Predictor agent.
- Action 2: Use only the local optimizer.
- Action 3: Use only the global optimizer.
- Action 4: Use the combination of Global and local optimizers without Predictor agent.

The above set of actions are constructed with two different 'Max' values of the Nelder-Mead thus constituting to a total set of 8 discrete actions. The graphical illustration of the action space is depicted in Figure 6.

**3) Reward:** Because the Environment has multiple optimizers for which the learning has to be done, the reward $R$ is determined using a *hybrid reward mechanism* to guide the agent's learning process. For the local and global optimizers, a positive reward is granted when an optimizer improves upon the best solution from the previous cycle; otherwise, no reward is given. The predictor agent is rewarded based on its prediction accuracy, with higher accuracy resulting in a higher reward. However, if the predictor incorrectly discards a feasible UGV-UAV solution, mistaking it for infeasible, a penalty is imposed to discourage such errors. Additionally, rewards account for computational efficiency, with each action evaluated for its computational time to encourage faster solutions. To prevent overfitting and encourage exploration, action masking is applied, assigning a significant penalty $L$ for repeated actions in succession. This approach discourages repetitive actions and promotes broader exploration, helping the agent converge towards an optimal policy.

**4) RL Policy: Proximal Policy Optimization** (PPO) is a policy-gradient method that works on Actor-Critic style which updates policies with a clipped objective function that helps to balance the exploration and exploitation while preventing larger policy updates. Since the nature of DRL implementation in this study is compuationally expensive, PPO is utilized because of its nature of simplicity and sample efficient [32].

**Policy Objective Function:** The objective in PPO is designed to maximize the expected reward while constraining the policy update to avoid drastic changes. This is achieved using the *clipped surrogate objective*, defined as:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[ \min \left( r_t(\theta)\hat{A}_t,\ \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{3.12}$$

where $r_t(\theta)$ is the probability ratio between the new and old policies:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{3.13}$$

In this equation:

- $\pi_\theta(a_t|s_t)$ is the probability of action $a_t$ under the current policy $\theta$ given the state $s_t$.
- $\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the probability of the same action under the previous policy.

These $\pi_\theta(a_t|s_t)$ is basically a Neural Network that acts as the function approximation to provide actions as output for the given current state as input. This neural network is learnt via the DRL framework using an appropriate learning rate that fits the computational expense of the problem considered.

The clipping function, $\text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)$, restricts the value of $r_t(\theta)$ within the interval $[1-\epsilon, 1+\epsilon]$, where $\epsilon$ is a hyperparameter (0.2 in this study) that controls the extent of the policy change. The advantage function $\hat{A}_t$ is used as an estimate of the advantage of taking action $a_t$ in state $s_t$.

### 3.3.5 Advantage Estimation

Advantage is a measure of reward that indicates how well a chosen action has performed relative to its expected return. The advantage function $\hat{A}_t$ is estimated through the temporal difference (TD) error:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \tag{3.14}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, with $\gamma$ representing the discount factor, and $\lambda$ a parameter for balancing bias-variance in advantage estimation through Generalized Advantage Estimation (GAE).

### 3.3.6 Value Function Loss

PPO uses an additional loss term for the value function to improve the critic's accuracy. The value function loss is typically defined as:

$$L^{\mathrm{VF}}(\theta) = \mathbb{E}_t\left[\left(V_\theta(s_t) - V_t^{\mathrm{target}}\right)^2\right] \tag{3.15}$$

where $V_t^{\mathrm{target}}$ represents the target value derived from the observed rewards.

### 3.3.7 Total Objective Function

Combining the clipped policy objective and the value function loss, the final objective function becomes:

$$L(\theta) = \mathbb{E}_t\left[L^{\mathrm{CLIP}}(\theta) - c_1 L^{\mathrm{VF}}(\theta) + c_2 S[\pi_\theta](s_t)\right] \tag{3.16}$$

where $c_1$ and $c_2$ are coefficients for the value function and entropy regularization terms, respectively, and $S(\pi_\theta)$ represents the entropy of the policy, encouraging exploration by preventing premature convergence.

### 3.3.8 Neural Network Architecture

The Policy function $\pi_\theta(a_t|s_t)$ and Value function $V_\theta(s_t)$ are Neural Networks and each of these Neural Networks are Multi-Layer Perceptron (MLPs) with two hidden layers and each layer has the size of 64 neurons. The activation function used throughout the network is the hyperbolic tangent function $Tanh$. The input and output layer of the Neural Network depends upon the dimensions of the input and output. In this case, the input layer for policy function would have 13 neurons and output layer would have 8 neurons.

The algorithm for PPO is described in Algorithm 1

In this study, $T$ represents a single episode length. The limit for $T$ was kept to be 4 to run the entire optimization cycle up to 4 iterations for one episode. The number of epochs $K$ was kept to be 4, $N$ to be 16 and mini batch size $M$ to be 8.

After each iteration, the policy receives a reward through a hybrid reward mechanism. This reward is an aggregate based on the individual contributions of each algorithm involved in the current step of the optimization. By continuously learning from this feedback, the RL-assisted framework adapts its selection of algorithms

**Algorithm 1** PPO algorithm

**for** iteration = 1, 2, ... **do**
  **for** actor = 1, 2, ..., N **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for $T$ timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  **end for**
  Optimize surrogate $L$ with respect to $\theta$, with $K$ epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
**end for**

to improve optimization efficiency and outcome quality across iterations. This structured, intelligent decision-making process ensures that the most appropriate subset of algorithms is chosen to improve the efficiency of the computation.

# 4 RESULTS

We used Python 3.10 for all the computations: the OpenAI's 'Stable Baselines 3' Reinforcement Learning (RL) package is used for RL agent, the Ensemble of classification algorithms like kNN, SVM, Decision Trees used in the Predictor agent is from the Scikit-learn package, a customized Genetic Algorithm, and Nelder-Mead from the Scipy package for performing UGV parameterized route optimization; and OR-Tools solver for UAV optimization. All computations are done on a 3.7 GHz Intel Core i9 processor with 32 GB RAM on a 64-bit operating system.

The scenarios consider varying number of UAVs and one UGV, with each UAV having a battery capacity of 4000 mAh (total energy of $E_a = 287.7kJ$) and the UGV energy capacity of $E_g = 25.01MJ$. The UAV and UGV are set to travel at velocities of $v_a = 10$m/s and $v_g = 4.5$m/s, respectively. The UAV follows the power consumption curve of $P_a = 0.0461(v_a)^3 - 0.5834(v_a)^2 - 1.8761v_a + 229.6$ and UGV follows the power curve $P_g = 464.8v_g + 356.3$ (referred from [33]). Based upon the scale of scenarios considered ($14 \times 14$ km), solving the Minimum Set Cover (MSC) problem results in a set of multiple candidate UGV routes $X_s$ with two primary rendezvous locations ($j = 2$) and three mid UGV-UAV rendezvous locations ($k = 3$) as the parameters for optimization. Thus, the UGV route optimization is parameterized by a total of ($\mathcal{S}$=5) decision variables, which are optimized within the bi-level optimization framework.

The Reinforcement Learning policy training is performed for several scenarios sent in batches. The nature of the scenario is of several task points situated in a patterned format with three branches meeting at a junction point. An example training scenario is shown in Figure 7. Keeping the pattern similar, the RL model is trained with 200 scenarios. Two different Learning Rates LR = 0.005 and LR = 0.001 were considered for training. The policy training curves for different learning rates can be seen in Figure 8. When the model was trained with LR greater than 0.005, it performed poorly due to the higher learning rate causing unstable learning, and hence LR parameter tuning was stopped. After analyzing training for different learning rates, the RL model with LR = 0.001 was chosen for testing. The training time with LR = 0.001 was found to be around 30 hours due to the inherent complexity in the learning mechanism of the

RL policy. For persistent surveillance, the end time of the mission $T$ is kept at 250 minutes.

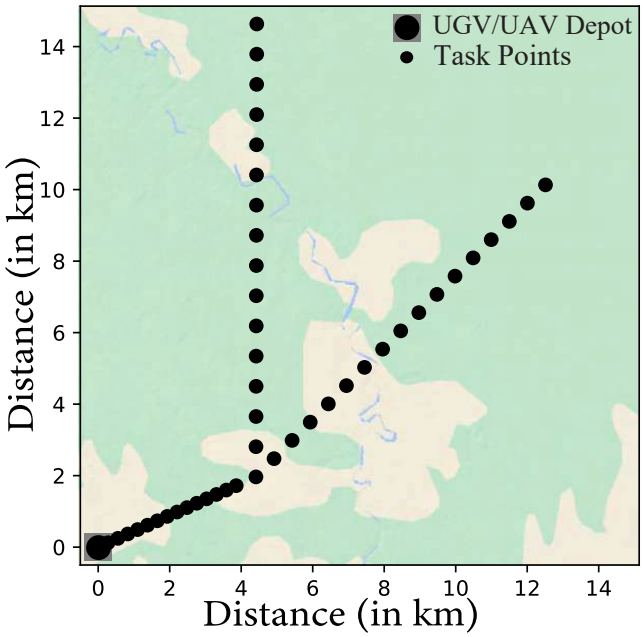



**Fig. 7:** An instance of training scenario

Extensive simulation analyses have been conducted to demonstrate the generalizability and efficiency of the proposed Reinforcement Learning-based A-Teams optimization framework. Various studies were performed to evaluate its performance. The trained RL model was compared against existing optimization frameworks, including an A-Teams variant, the conventional A-Teams approach, and Genetic Algorithms, to assess both objective value and computational time. One particular study examined the impact of task point density within a 16 km x 16 km scenario, categorizing the density into three levels: densely spaced, moderately spaced, and sparsely spaced. In densely spaced scenarios, task points are clustered closely together, while in moderately spaced scenarios, task points are more spread out. In sparsely spaced scenarios, the task points are distributed with even greater separation. A graphical representation of the different task point distribution for testing the optimization model is shown in Figure 9. This study considers a single UAV-UGV and multi UAV-UGV instances and the results are compared. A case study scenario was considered where a certain set of random points appear suddenly and the optimizer re-plan the UAV-UGV routes considering the dynamic changes into account to assess the ability of the proposed method to handle such changes.

There are four different methods being compared for the optimality of the solution and the computational time took to solve. One is the proposed RL-assisted A-Teams,

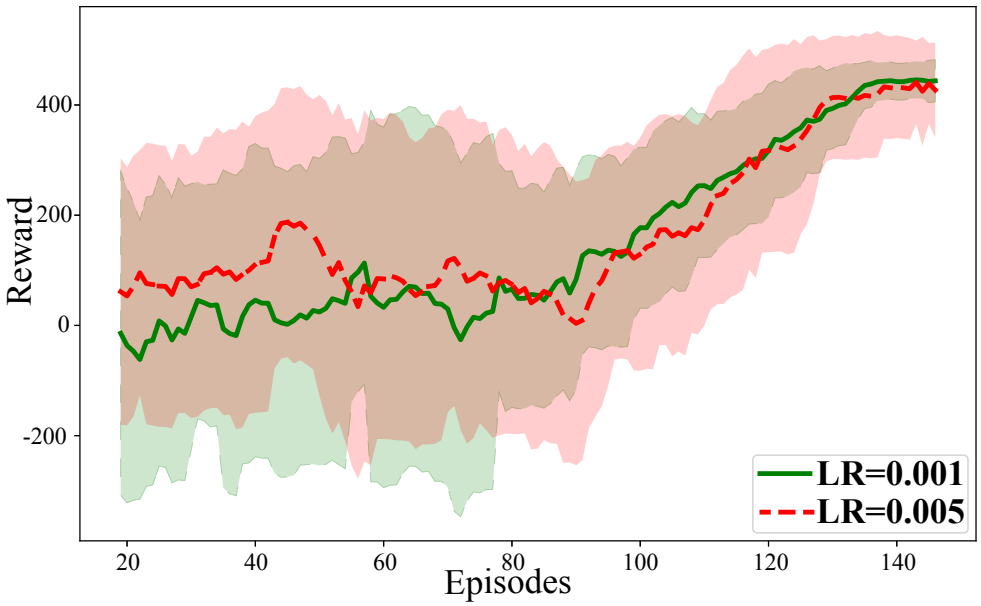



**Fig. 8:** RL Training with different Learning Rates (LR). The green and red bands correspond to 90% confidence band of the reward values for learning rates 0.001 and 0.005.

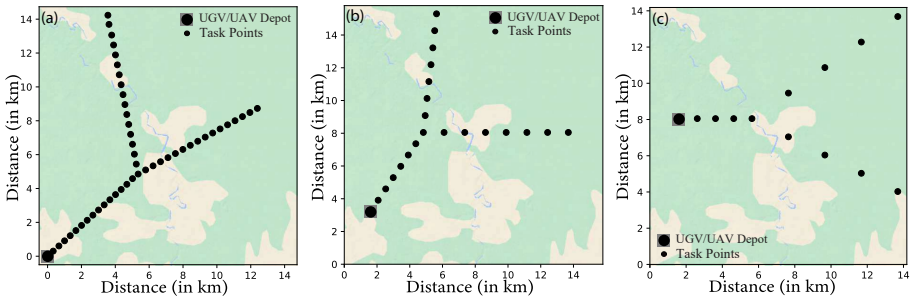



**Fig. 9:** Example of different task point distribution scenario types. a) High Dense scenario b) Moderate Dense scenario c) Sparse Dense scenario.

another method is A-Teams variant, the third method is original A-Teams and the fourth method is Genetic Algorithm. All these methods varies the way UGV optimization is performed at the outer-level. At the inner-level, the UAV optimization is performed by solving the E-VRP for all these methods. These details are found from the Tables 1 through 3.

**Table 1:** Comparison across different optimization methods for high-density task-point distribution with 1 UAV–1 UGV routing.

| Scenario # | RL-Assisted A-Teams | | A-Teams Variant | | Original A-Teams | | Genetic Algorithm | |
|---|---|---|---|---|---|---|---|---|
| | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) |
| 1 | 12 | 375 | 18 | 407 | 18 | 407 | 40 | 375 |
| 2 | 13 | 223.5 | 18 | 223.5 | 19 | 223.5 | 46 | 280.97 |
| 3 | 12 | 333 | 17 | 333 | 19 | 333 | 53 | 333 |
| 4 | 14 | 303 | 17 | 303 | 19 | 303 | 45 | 303 |

**Table 2:** Comparison across different optimization methods for high-density task-point distribution with 1 UAV–1 UGV routing.

| Scenario # | RL-Assisted A-Teams | | A-Teams Variant | | Original A-Teams | | Genetic Algorithm | |
|---|---|---|---|---|---|---|---|---|
| | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) |
| 1 | 9 | 116.81 | 10 | 122.76 | 12 | 122.76 | 27 | 122.76 |
| 2 | 7 | 106.07 | 8 | 106.07 | 8 | 106.07 | 25 | 106.07 |
| 3 | 10 | 113.9 | 12 | 119.14 | 13 | 117.4 | 33 | 101.34 |
| 4 | 8 | 136.82 | 10 | 136.82 | 11 | 136.82 | 26 | 120.97 |

## 4.1 Study 1: Comparison of different methods across various task point distributions with 1 UAV and 1 UGV routing

From Figure 10, the results demonstrate that across various task point distribution scenarios, RL-assisted A-Teams achieved computationally efficient optimization while maintaining the quality of the optimal solutions. Notably, in high-density task point distributions, the reduction in computational time was significantly greater compared to other methods. This is because the outer-level optimization focuses on optimizing UGV route parameters, which include the primary and mid-rendezvous locations along the branches of the task points. In high-density scenarios, the larger number of task points results in a greater number of UGV route parameter combinations, expanding the search space. Consequently, the RL policy has more opportunities to enhance optimization, leading to improved computational efficiency. Conversely, in low-density task point distributions, where fewer task points exist, the number of potential UGV route parameter combinations is reduced, which limits the ability of the RL-assisted

**Table 3:** Comparison across different optimization methods for high-density task-point distribution with 1 UAV–1 UGV routing.

| Scenario # | RL-Assisted A-Teams | | A-Teams Variant | | Original A-Teams | | Genetic Algorithm | |
|---|---|---|---|---|---|---|---|---|
| | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) |
| 1 | 6 | 79.1 | 5 | 83.5 | 5 | 79.1 | 23 | 83.5 |
| 2 | 4 | 91.5 | 5 | 91.5 | 5 | 91.5 | 13 | 91.5 |
| 3 | 5 | 84.2 | 5 | 84.2 | 5 | 84.2 | 22 | 84.2 |
| 4 | 4 | 77.5 | 4 | 77.5 | 4 | 77.5 | 23 | 77.5 |

A-Teams to significantly contribute to optimization improvements. The route sequence visualization of one of the scenarios with 1 UAV - 1 UGV routing is shown in Figure 11. Also from Figure 10, it can be seen that the proposed A-Teams with RL policy decision maker performs well computationally as compared to other A-Teams approaches, which demonstrates the capability of using RL based hyperheuristics for solving this collaborative UAV-UGV routing.

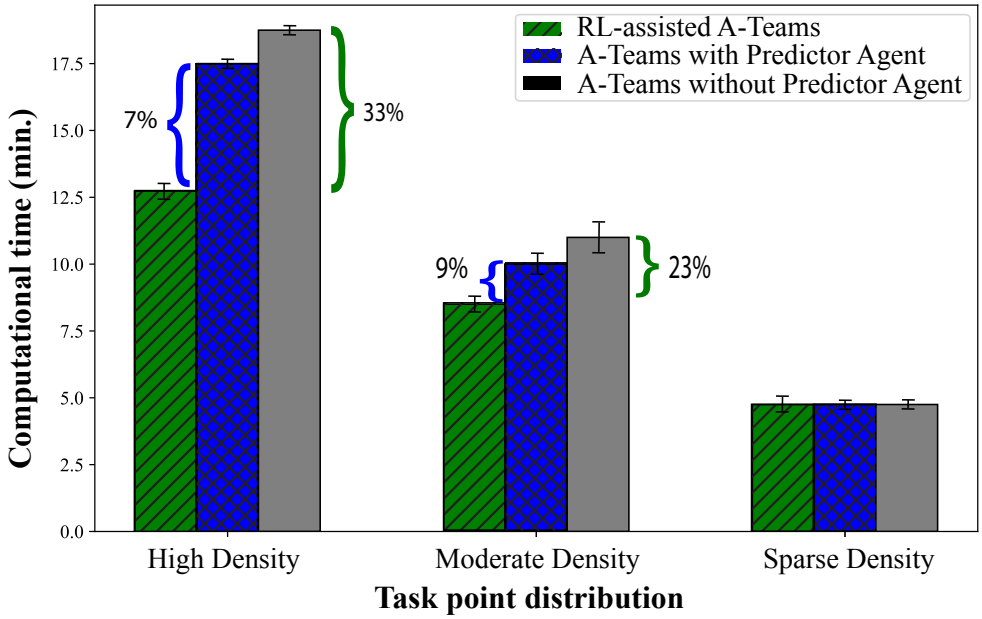



**Fig. 10:** Computational time comparison across different methods: For 1 UAV - 1 UGV study

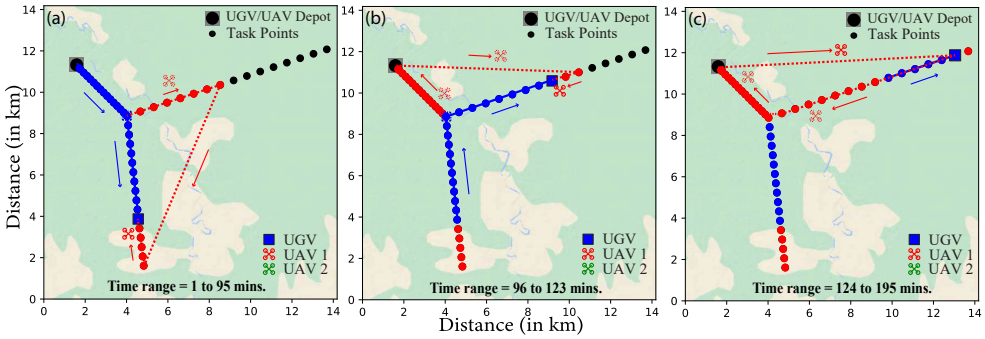



**Fig. 11:** Route sequence of Scenario 1 of High Dense distribution with 1 UAV - 1 UGV routing

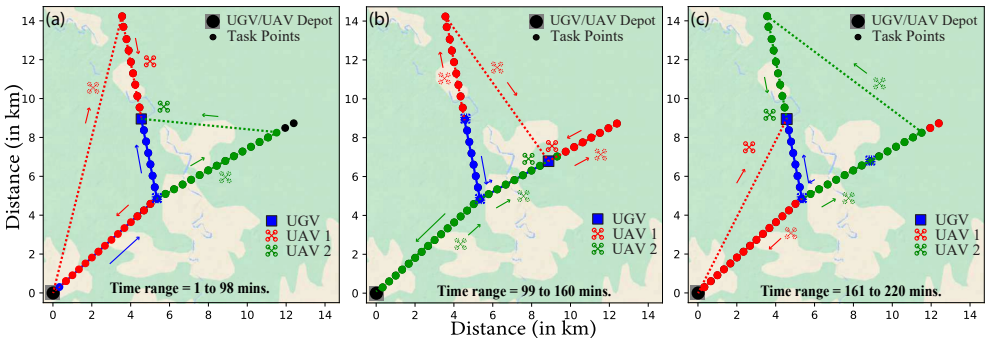



**Fig. 12:** Route sequence of Scenario 1 of High Dense distribution with 2 UAV - 1 UGV routing

## 4.2 Study 2: Comparison of different methods across various task point distributions with multi-UAV and 1 UGV routing

In order to study the generalizability, these methods are extended to perform persistent surveillance for multi-UAV UGV system. Specifically in this study, the 2 UAV - 1 UGV system is considered across the various task point distributions. In case of multi-UAV UGV routing, it is assumed that both the UAVs would start and land at the same location along the UGV's travel where the UGV waits to collect the UAVs for recharging.

Similar to the Study 1, optimality of the solution and the computational time took to solve are compared against the 4 methods implemented. These details are found in Tables 4 through 6. From the Tables and Figure 13, it can be seen that the trend is similar to the 1 UAV - 1 UGV routing problem, but the 2-UAV UGV routing results have a higher reduction in computational time when the proposed method is implemented against the existing methods. Moreover, comparing the overall

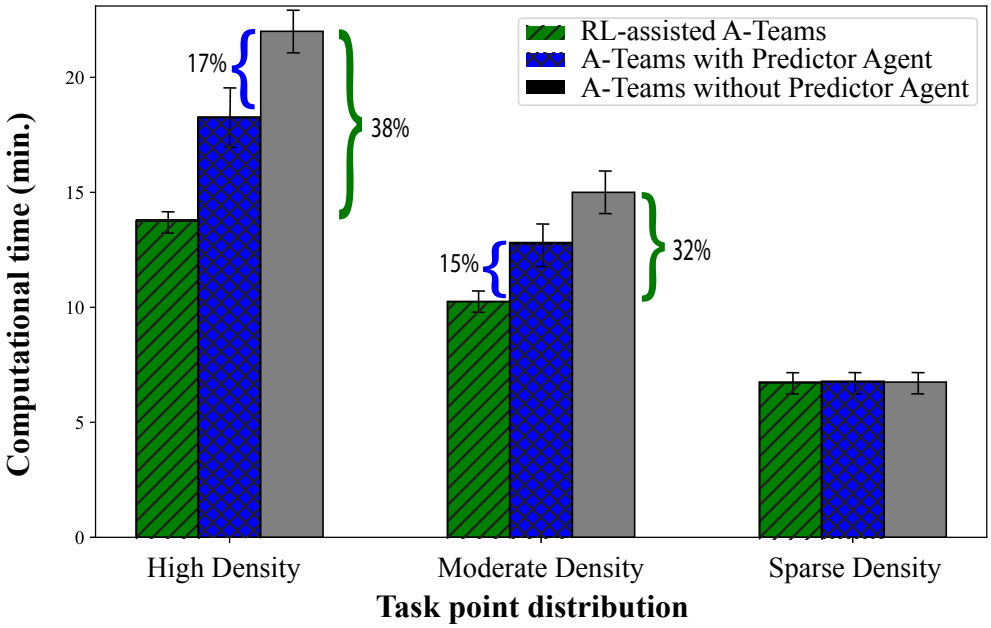



**Fig. 13:** Computational time comparison across different methods: For 2 UAV - 1 UGV study

results between 1 UAV-UGV system and 2 UAV-UGV system, it can be seen that the latter system leads to reduction in the objective function value for that corresponding Persistent surveillance mission time. This is because using multiple UAVs can help in visiting the task points more frequently, and hence, there is a greater reduction in the age period between visits. The route sequence visualization of one of the scenarios with 2 UAV - 1 UGV routing is shown in Figure 12. Also from Figure 13, it can be seen that

**Table 4:** Comparison across different optimization methods for high-density task-point distribution with 2 UAV–1 UGV routing.

| Scenario # | RL-Assisted A-Teams | | A-Teams Variant | | Original A-Teams | | Genetic Algorithm | |
|---|---|---|---|---|---|---|---|---|
| | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) |
| 1 | 13 | 281.2 | 19 | 281.2 | 22 | 212.9 | 45 | 231.6 |
| 2 | 16 | 231.4 | 24 | 231.4 | 26 | 231.4 | 62 | 209.8 |
| 3 | 13 | 268.6 | 15 | 268.6 | 20 | 254.6 | 44 | 254.6 |
| 4 | 13 | 281 | 15 | 281 | 20 | 281 | 46 | 281 |

**Table 5:** Comparison across different optimization methods for high-density task-point distribution with 2 UAV–1 UGV routing.

| Scenario # | RL-Assisted A-Teams | | A-Teams Variant | | Original A-Teams | | Genetic Algorithm | |
|---|---|---|---|---|---|---|---|---|
| | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) |
| 1 | 11 | 107.3 | 15 | 107.3 | 17 | 107.3 | 30 | 104.3 |
| 2 | 11 | 84.3 | 13 | 102.5 | 15 | 84.28 | 29 | 101.2 |
| 3 | 11 | 91.9 | 14 | 91.9 | 17 | 94.23 | 40 | 91.9 |
| 4 | 8 | 101.5 | 9 | 101.5 | 11 | 101.5 | 30 | 101.5 |

**Table 6:** Comparison across different optimization methods for high-density task-point distribution with 2 UAV–1 UGV routing.

| Scenario # | RL-Assisted A-Teams | | A-Teams Variant | | Original A-Teams | | Genetic Algorithm | |
|---|---|---|---|---|---|---|---|---|
| | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) | Objective | Computing Time (min.) |
| 1 | 7 | 69.7 | 7 | 69.7 | 7 | 69.7 | 25 | 69.7 |
| 2 | 7 | 78.8 | 7 | 78.8 | 7 | 78.8 | 28 | 78.8 |
| 3 | 8 | 80.5 | 8 | 80.5 | 8 | 80.5 | 27 | 80.5 |
| 4 | 5 | 48.5 | 6 | 48.5 | 5 | 48.5 | 26 | 48.5 |

the proposed A-Teams with RL policy decision maker performs well computationally as compared to other A-Teams approaches, which demonstrates the capability of using RL based hyperheuristics for solving this collaborative UAV-UGV routing generalized to multi-UAV UGV systems.

## 4.3 Study 3: Case Study - Bridge Inspection Using Autonomous Vehicle System

From the above results, it is seen that the proposed Reinforcement Learning method solves the cooperative UAV-UGV routing problem with more computational efficiency as opposed to other methods. To evaluate the real-world applicability of the proposed optimization method and harness the potential of implementing autonomous UAV-UGV vehicle team on a civilian application, the trained RL-Assisted A-Teams

framework is implemented on a case study based on Bridge Inspection. Bridge inspection is a critical task to ensure structural safety and prevent catastrophic failures. Traditional inspection methods are often time-consuming, costly, and hazardous for inspectors. Autonomous vehicles offer a safer, faster and more cost-effective alternative to infrastructure monitoring. Some of the works in the literature by Chan et al. [34] and Koch et al. [35] highlight the advantages of drone-based inspection over conventional methods, including cost reduction, faster inspection, and less inconvenience to travelers in the road. Typically, in these applications, drones are used for visual inspections by taking photos of the intersection point between the bridge deck and the columns. In particular, Omar et al. [36] investigate the use of drones equipped with infrared thermography to detect subsurface delamination on concrete bridge decks, offering a noncontact and non-disruptive inspection method. Unlike traditional approaches that rely solely on visual assessment, drones equipped with thermal cameras can identify hidden structural defects by capturing temperature variations across the deck surface. This shows that drones are also capable of performing an over-the-deck inspection. Other applications include combining drones with manually operated vehicles for bridge inspection. In this approach, drones inspect the intersection points of the deck and columns, while a car or truck equipped with Ground Penetrating Radar (GPR) conducts surface and subsurface inspections of the deck.

By implementing a UAV-UGV team, our study enables full-autonomous bridge inspection with minimal human intervention. This approach improves efficiency, reduces inspection time, and improves safety throughout the process.

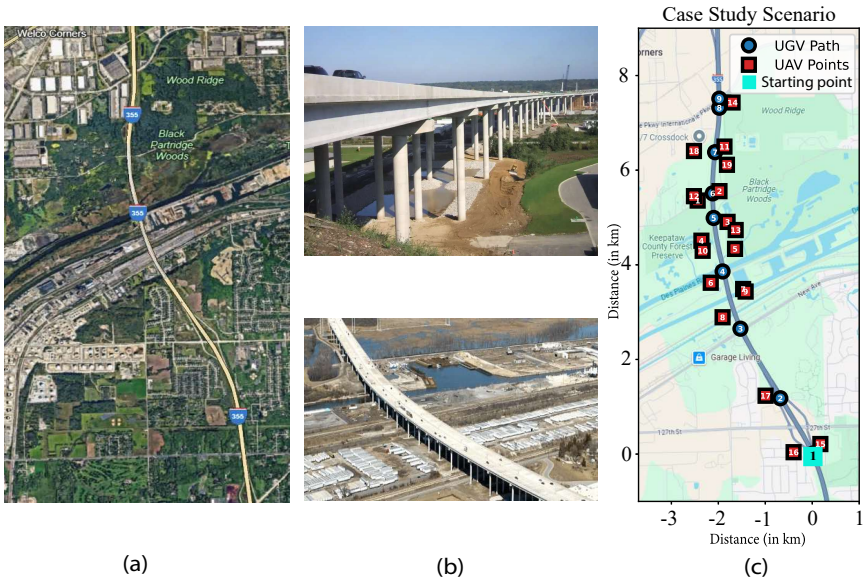



**Fig. 14:** Case Study scenario considered for Bridge inspection. a) Bridge locations obtained from ArcGIS map b) Aerial view of the Bridge c) Converted coordinates for performing routing as per cartesian coordinates.

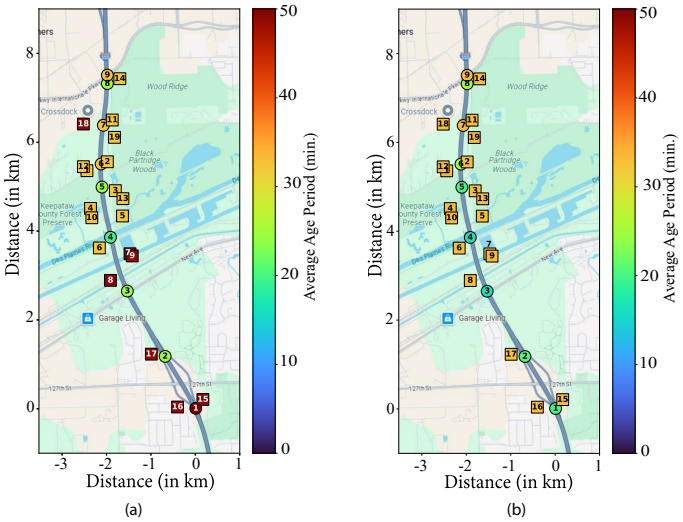



**Fig. 15:** Average age period heatmap. a) For 1 UAV - 1 UGV system. b) For 2 UAV - 1 UGV system.

For this case study, a bridge segment located near Chicago, Illinois, USA, was selected. Spanning approximately 8 kilometers, this segment comprises three interconnected bridges: the Long Run Creek Bridge, the Des Plaines River Valley Bridge, and the I-355 Bridge. Initial data regarding the bridge locations—including latitude and longitude—was obtained from ArcGIS mapping tools. This geospatial data was subsequently converted into Cartesian (x, y) coordinates for analysis.

Figure 14 provides a graphical overview of the case study. Fig. 14 (a) shows the original bridge locations as extracted from ArcGIS. Fig. Figure 14 (b) presents a photograph of the actual bridge structure. Fig, Figure 14 (c) depicts the transformed Cartesian coordinate representation of the bridge network. In this coordinate-based layout, blue circles denote UGV (Unmanned Ground Vehicle) traversal points, which are used for conducting over-the-bridge inspections. Red squares represent UAV (Unmanned Aerial Vehicle) waypoints, strategically placed near the intersections of bridge decks and columns to facilitate visual inspection.

To ensure a realistic simulation setup, the following specifications are adopted: the UAV operates at a speed of 10 m/s, based on the parameters used in [37], while the UGV moves at a speed of 2.5 m/s, reflecting typical Ground Penetrating Radar (GPR) traversal speeds as referenced in [38]. Additionally, since visual or infrared inspections typically require between 30 to 60 seconds per point, an inspection duration of 60 seconds is considered in this study.

A persistent surveillance operation is carried out over a span of 150 minutes. UAVs are assumed to be equipped with both RGB and infrared (IR) cameras, enabling them to perform both visual and thermal inspections of the bridge. This dual capability enhances the adaptability and effectiveness of the system for autonomous bridge inspection tasks.

To evaluate the effectiveness of autonomous vehicle team configurations, simulations were performed using varying numbers of UAVs while keeping the number of UGVs constant. Specifically, a comparison was made between a 1 UAV – 1 UGV system and a 2 UAV – 1 UGV system, both deployed for persistent surveillance over a period of 150 minutes. The performance metric used for this comparison is the Age Period of the task points, which reflects the time elapsed since each point was last visited. Figure 15 presents a heatmap of the age period for the task points visited by each team configuration. The results clearly show that the 2 UAV – 1 UGV system outperforms the 1 UAV – 1 UGV setup, achieving consistently lower age periods across all nodes. This indicates a more frequent and efficient coverage of inspection points. Our earlier work in [39] also supports this finding by demonstrating that a UAV-to-UGV ratio of 2:1 yields more efficient routing and inspection performance. Consistent with that insight, the 2 UAV – 1 UGV configuration used here adheres to the 2:1 ratio, resulting in improved inspection efficiency compared to the 1:1 configuration.

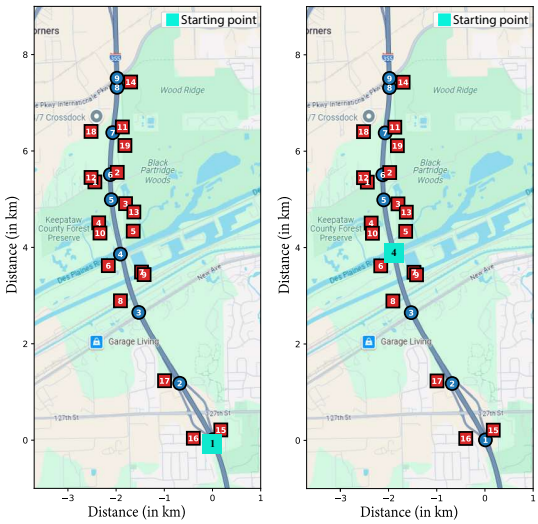



**Fig. 16:** Case Study scenario with different starting points. The Green box in (a) and (b) represents the location from where the UAV-UGV inspection starts.

To test the UGV-UAV routing across different starting points to see the effect of routing, we considered two starting locations. One case is to start the vehicle system from the bottom point and the other case is to start them from the middle of the bridge. Figures 16 a) and b) show the different starting points. The routing comparison is made between these two cases for 1 UAV-UGV and 2 UAV-1UAV systems. Table 7 shows the average value of the average age period of all nodes for the respective vehicle system and the different starting points. From the table it can be seen that, while the multi-UAV-UGV system reduces the age period overall with a reduction of

more than 7 minutes on average, the cases with bottom starting point perform better for both the vehicle system than starting from the middle.

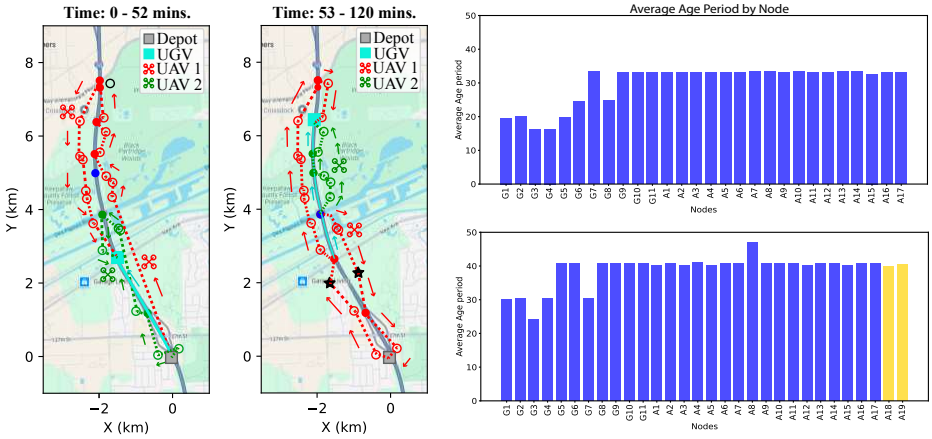



**Fig. 17:** (a) Case Study scenario routing sequence. Black stars on the right plot represent the dynamic appearance of inspection points to be visited. (b) Average age period comparison across all the nodes. Top plot represent the 2 UAV - 1 UGV routing without dynamic changes and bottom plot represents the 2 UAV - 1 UGV routing with dynamic changes. The yellow bars represent the additionally added points (Black stars from Figure (a)).

**Table 7:** Average value of Average Age period across all nodes across different vehicle system vs different starting point.

| UAV-UGV system # | Bottom-start routing (in mins.) | Mid-start routing (in mins.) |
|---|---|---|
| 1 UAV - 1 UGV | 37 | 39 |
| 2 UAV - 1 UGV | 30 | 33 |

To further evaluate the efficiency of the optimization of the proposed method, dynamic planning was incorporated into the persistent surveillance framework, specifically for scenarios where new mission points emerge randomly during the routing process. This dynamic planning approach was tested in a 2 UAV - 1 UGV system, where the newly introduced inspection points were assumed to be located outside the road network, representing unexpected visual inspections of the bridge columns by UAVs. A key assumption in this study is that UAVs become aware of dynamic mission

updates during their rendezvous with the UGV. The UGV serves as the primary communication node, relaying new mission points whenever it comes into contact with a UAV. Figure 17 (a) illustrates the route plan that incorporates dynamic changes, where stars indicate the dynamically introduced points that appeared during the timestamp (enter the appropriate timestamp here). These points are transmitted to the UAVs during their rendezvous with the UGV. During the recharging phase, the optimization method re-plans the UAV routes to accommodate the dynamic changes. Before takeoff, UAVs receive updated route assignments, ensuring that newly introduced inspection points are integrated into the mission plan. The average re-optimization time was found to be 3 minutes, which is significantly faster than the 15-minute UAV recharging duration, demonstrating the feasibility and efficiency of real-time re-planning in dynamic environments. Figure 17 (b) represents the average age period of each node in the scenario for the 2 UAV - 1 UGV system. The top plot represents the scenario routing without dynamic changes and the bottom plot represents the routing with dynamic changes. The results indicate that the inclusion of dynamic mission points increases the overall average age period of each node. This is attributed to the additional visits required by the UAV-UGV system to accommodate the newly introduced inspection tasks.

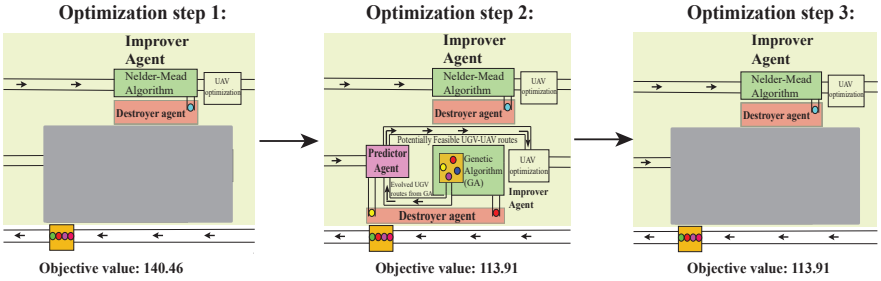



**Fig. 18:** RL Policy-Driven Optimization Sequence for Agent Selection. Each Optimization step corresponds to performing one iteration of the bi-level optimization.

# 5 DISCUSSION

This research work develops a Reinforcement Learning assisted A-Teams optimization framework that is computationally efficient to solve for optimal UAV-UGV routes quickly. The capability is also demonstrated by considering different test scenarios with 1 UAV-UGV system, 2 UAV-UGV system and scenario with dyamic changes. The key to computational efficiency from implementing the RL policy as a decision maker in strategically selecting which algorithms to take part in what step of the optimization process. This strategic selection helps to impart computational efficiency because, in each step of the optimization process, the RL policy chooses the action (in this case, the optimization algorithm) that maximizes the expected reward, which in this case, if the computational time is significantly less, then those action sequences are

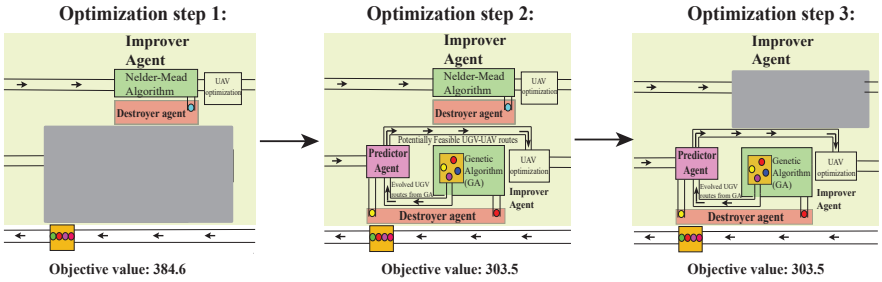



**Fig. 19:** RL Policy-Driven Optimization Sequence for Agent Selection. Here Alternative reward mechanism is shaped to encourage choosing the Predictor agent as a part of optimization process.

encouraged. From Figures 10 and 13 it can be seen that the proposed method achieves a computational efficiency of 33% at the most for 1 UAV - 1 UGV system, while it achieves a computational efficiency of 38% at the most for 2 UAV - 1 UGV system.

The A-Teams multi-agent framework enhances solution quality by integrating algorithms with complementary strengths, allowing for a streamlined optimization process. For instance, pairing a global search algorithm like Genetic Algorithm (GA) with a local search method such as Nelder-Mead accelerates the search for near-optimal solutions, outperforming GA alone. Constraint Programming (CP) for UAV routing further contributes to computational efficiency by leveraging heuristic-based optimization. Additionally, the Predictor Agent employs an ensemble model (k-NN, SVM, Decision Trees) to assess the feasibility of UGV-UAV routes, making balanced predictions from multiple classifiers to discard potentially infeasible solutions.

The RL policy acts as a high-level decision-maker, selecting the most suitable algorithm at each step to maximize the expected reward, where actions correspond to the available optimization algorithms. By strategically choosing algorithms, the RL agent ensures that UGV route parameters are optimized in a way that minimizes unnecessary UAV evaluations. This targeted selection minimizes the search time while maintaining solution quality, as the RL policy carefully selects the choice of algorithms to achieve efficient computation throughout the optimization process.

To obtain an effective RL policy in this framework lies in reward shaping. Given that the environment is a multi-agent optimization framework, a hybrid reward mechanism is crucial, as it guides the policy training to align with the optimization goal. When computational efficiency is prioritized, actions with lower computation times are weighted more heavily in the reward structure. This approach is illustrated in Figure 18, which shows an optimization scenario using the RL-Assisted A-Teams method for a moderate-density task point distribution (Scenario 3) with a 1 UAV - 1 UGV system.

In this example, the policy initially selects only the local optimizer to refine the current best solution generated by the constructor. In the following step, it combines both the local (Nelder-Mead) and global optimizers (GA) without the Predictor Agent to drive further improvement. After the solution improves, the policy reverts to the local optimizer alone in the third step to determine whether the enhanced solution can be further refined. The process terminates when no further improvement is found.

The policy could choose this sequence because, in this study's framework, the local optimizer requires fewer function evaluations than the global optimizer. By focusing on local improvement, it can quickly identify when no further gains are achievable, thus terminating faster than the Genetic Algorithm, which explores the broader search space. This selective use of optimization algorithms minimizes unnecessary evaluations and enhances computational efficiency.

When reward shaping prioritizes the inclusion of the Predictor Agent in the optimization process, the policy learns to find an optimization sequence that minimizes computational time while preserving solution quality, specifically by encouraging the selection of actions that involve the Predictor Agent. Figure 19 shows an optimization sequence for Scenario 4 of high density task distribution, where the RL policy is trained using an alternative reward mechanism that gives more weightage to choosing an action by utilizing the Predictor Agent as a part of the optimization process by encouraging that action. The computational time took to solve through this method was 16 minutes. Referring to the Table 1 in the 4th row, it can be seen that it performs fairly competitive with other methods whilst retaining the quality of the solution.

This example highlights the pivotal role of reward shaping in training an effective RL policy, allowing adjustments based on specific optimization objectives.

The proposed method comes with some limitations. Since the RL policy used in this work is a learning-based hyperheuristics, it only acts at the higher level. Lower-level UAV optimization is computationally expensive and significantly slows down training despite using the sample-efficient PPO algorithm. This limits the scalability of the method to larger datasets, more complex problems with additional hyperparameters, and frameworks with many algorithms, where the expanded action space further complicates the learning of an effective RL policy. Additionally, the learned RL model can be applied only to the type of scenarios being considered during training, limiting its ability to generalize across different types of task point distribution considered. Another limitation is that typically, the interpretability of the RL method is hard, limiting the overall understanding of the actions chosen by the policy that has been learned. We have proposed another method for performing multi UAV-UGV optimization using Reinforcement Learning where the RL is used directly at the lower-level to perform the UAV and UGV route optimization [40].

In the context of Vehicle Routing Problem, even though this method aims to reduce the computational time by efficiently searching the optimization space, it may still yield suboptimal results compared to the mathematical programming methods, as it still depends upon the metaheuristics. Finally, the effectiveness of A-Teams optimization relies on its associated agents. For instance, the quality of the initial solution pool generated by the Constructor agent greatly influences the search direction, and the Predictor agent's ML model must achieve a high F-score for reliable classification.

# 6 CONCLUSION

This study proposes a Reinforcement Learning-based hyperheuristic as a high-level decision-maker within the multi-agent A-Teams framework to enhance optimization

efficiency. The method is benchmarked against various A-Teams variants and a standard Genetic Algorithm. Results demonstrate that the proposed approach consistently generates high-quality heuristic solutions in significantly less computational time compared to the baseline methods, for both single (1 UAV–UGV) and multi-agent (2 UAV–UGV) systems. To assess robustness, the framework was validated across a range of scenario instances, including practical simulations under dynamic environmental conditions.

Furthermore, the adoption of a 2 UAV–UGV system notably reduces the task point age-period, i.e., the time since last visit, due to the increased frequency of coverage enabled by multiple UAVs. This outcome aligns with prior studies emphasizing the advantages of multi-agent systems for persistent surveillance tasks, thereby reinforcing the practical effectiveness of the proposed RL-based framework.

## Acknowledgements

The authors would like to express their gratitude to the DEVCOM Army Research Laboratory for their financial support in funding this project under grant W911NF-14-S-0018.

The authors would also like to thank Prof. Didem Ozevin, Professor, University of Illinois Chicago (dozevin@uic.edu) for her valuable insights on practical applications of bridge inspection, which greatly helped in structuring the case study for this paper.

## Declarations

- **Funding** The work is funded by Army Research Laboratory grant W911NF-14-S-003
- **Conflict of Interests** The authors declare no conflict of interest
- **Ethics Approval** Not applicable
- **Consent to Participate** Not applicable
- **Consent for Publication** Not applicable
- **Code or data availability** The data and code availability can be found in the following Github page: https://github.com/Subram0212/RL-Assisted-A-Teams. Currently this is a private repository, however it can be obtained upon a credible request.
- **Authors' contributions** Subramanian Ramasamy (SR), Md Safwan Mondal (SM), James D. Humann (JH), James M. Dotterweich (JD), Pranav Bhounsule (PB). Conceptualization: SR, SM, JH, JD, PB; Methodology: SR, SM and PB; Software and validation: SR; Writing—original draft preparation: SR; Writing—Review and editing: SM, JH, JD, PB; Supervision: JH, JD, PB; Project administration: PB. All authors have read and agreed to the published version of the manuscript.

## References

[1] Pratap Tokekar, Joshua Vander Hook, David Mulla, and Volkan Isler. Sensor planning for a symbiotic uav and ugv system for precision agriculture. *IEEE transactions on robotics*, 32(6):1498–1511, 2016.

[2] 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.

[3] Roberto Baldacci, Maria Battarra, and Daniele Vigo. Routing a heterogeneous fleet of vehicles. *The vehicle routing problem: latest advances and new challenges*, pages 3–27, 2008.

[4] 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.

[5] Wei Gao, Junren Luo, Wanpeng Zhang, Weilin Yuan, and Zhiyong Liao. Commanding cooperative ugv-uav with nested vehicle routing for emergency resource delivery. *IEEE Access*, 8:215691–215704, 2020.

[6] Sepehr Seyedi, Yasin Yazicioğlu, and Derya Aksaray. Persistent surveillance with energy-constrained uavs and mobile charging stations. *IFAC-PapersOnLine*, 52(20):193–198, 2019.

[7] Christian Blum, Jakob Puchinger, Günther R. Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.

[8] Jianjun Ni, Xiaotian Wang, Min Tang, Weidong Cao, Pengfei Shi, and Simon X. Yang. An improved real-time path planning method based on dragonfly algorithm for heterogeneous multi-robot system. *IEEE Access*, 8:140558–140568, 2020.

[9] Ning Ma, Yunfeng Cao, Xinyao Wang, Zhaoyang Wang, and Houjun Sun. A fast path re-planning method for uav based on improved a* algorithm. In *2020 3rd International Conference on Unmanned Systems (ICUS)*, pages 462–467, 2020.

[10] Simón Martínez-Rozas, Rafael Rey, David Alejo, Domingo Acedo, José Antonio Cobano, Alejandro Rodríguez-Ramos, Pascual Campoy, Luis Merino, and Fernando Caballero. An aerial/ground robot team for autonomous firefighting in urban gnss-denied scenarios. 2022.

[11] Piotr Jedrzejowicz and Ewa Ratajczak-Ropel. Experimental evaluation of a-teams solving resource availability cost problem. In *Intelligent Decision Technologies 2019: Proceedings of the 11th KES International Conference on Intelligent Decision Technologies (KES-IDT 2019), Volume 1*, pages 213–223. Springer, 2019.

[12] Michela Milano and Andrea Roli. Magma: a multiagent architecture for metaheuristics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(2):925–941, 2004.

[13] Subramanian Ramasamy, Md Safwan Mondal, James D Humann, James M Dotterweich, Jean-Paul F Reddinger, Marshal A Childers, and Pranav A Bhounsule. Computationally efficient multi-agent optimization framework for online routing of uav-ugv system. In *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, pages 204–211. IEEE, 2024.

[14] Nasser Lotfi and Adnan Acan. Learning-based multi-agent system for solving combinatorial optimization problems: A new architecture. In *Hybrid Artificial Intelligent Systems: 10th International Conference, HAIS 2015, Bilbao, Spain,*

*June 22-24, 2015, Proceedings 10*, pages 319–332. Springer, 2015.

[15] Peter Cowling, Graham Kendall, and Eric Soubeiga. A hyperheuristic approach to scheduling a sales summit. In *Practice and Theory of Automated Timetabling III: Third International Conference, PATAT 2000 Konstanz, Germany, August 16–18, 2000 Selected Papers 3*, pages 176–190. Springer, 2001.

[16] Peter Cowling, Graham Kendall, and Limin Han. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1185–1190. IEEE, 2002.

[17] Jacomine Grobler, Andries P Engelbrecht, Graham Kendall, and Venkata SS Yadavalli. Heuristic space diversity control for improved meta-hyper-heuristic performance. *Information Sciences*, 300:49–62, 2015.

[18] Nasser R Sabar, Masri Ayob, Graham Kendall, and Rong Qu. A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Cybernetics*, 45(2):306–315, 2015.

[19] Michail G Lagoudakis, Michael L Littman, et al. Algorithm selection using reinforcement learning. In *ICML*, pages 511–518, 2000.

[20] Warren Armstrong, Peter Christen, Eric McCreath, and Alistair P Rendell. Dynamic algorithm selection using reinforcement learning. In *2006 international workshop on integrating ai and data mining*, pages 18–25. IEEE, 2006.

[21] Rodrigo Garrido-Castro, Andreas Turke, and Tom van Woensel. A flexible and adaptive hyper-heuristic approach for (dynamic) capacitated vehicle routing problems. *European Journal of Operational Research*, 222(1):111–122, 2012.

[22] Yuan Yao, Zhe Peng, and Bin Xiao. Parallel hyper-heuristic algorithm for multi-objective route planning in a smart city. *IEEE Transactions on Vehicular Technology*, 67(11):10307–10318, 2018.

[23] Hadi Mosadegh, SMT Fatemi Ghomi, and Gürsel A Süer. Stochastic mixed-model assembly line sequencing problem: Mathematical modeling and q-learning based simulated annealing hyper-heuristics. *European Journal of Operational Research*, 282(2):530–544, 2020.

[24] 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.

[25] 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.

[26] C. E. Miller, A. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7:326–329, 1960.

[27] 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.

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

[29] Katharine Mullen, David Ardia, David L Gil, Donald Windover, and James Cline. Deoptim: An r package for global optimization by differential evolution. *Journal of Statistical Software*, 40(6):1–26, 2011.

[30] Sarosh Talukdar, VC Ramesh, et al. Cooperative methods for security planning. 1992.

[31] Subramanian Ramasamy, Md Safwan Mondal, Jean-Paul F Reddinger, James M Dotterweich, James D Humann, Marshal A Childers, and Pranav A Bhounsule. Solving vehicle routing problem for unmanned heterogeneous vehicle systems using asynchronous multi-agent architecture (a-teams). In *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 95–102. IEEE, 2023.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[33] 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.

[34] Brodie Chan, Hong Guan, Jun Jo, and Michael Blumenstein. Towards uav-based bridge inspection systems: A review and an application perspective. *Structural Monitoring and Maintenance*, 2(3):283–300, 2015.

[35] Christian Koch, S German Paal, Abbas Rashidi, Zhenhua Zhu, Markus König, and Ioannis Brilakis. Achievements and challenges in machine vision-based inspection of large concrete structures. *Advances in Structural Engineering*, 17(3):303–318, 2014.

[36] Tarek Omar and Moncef L. Nehdi. Remote sensing of concrete bridge decks using unmanned aerial vehicle infrared thermography. *Automation in Construction*, 83:360–371, 2017.

[37] Junwon Seo, Luis Duque, and Jim Wacker. Drone-enabled bridge inspection methodology and application. *Automation in construction*, 94:112–126, 2018.

[38] Infrasense. Principles of Ground Penetrating Radar for Bridge Deck Inspection. https://infrasense.com/gpr-bridge-deck/?utm_source=chatgpt.com, 2025. Online; accessed Mar 12, 2025.

[39] 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.

[40] Md Safwan Mondal, Subramanian Ramasamy, James D Humann, James M Dotterweich, Jean-Paul F Reddinger, Marshal A Childers, and Pranav Bhounsule. An attention-aware deep reinforcement learning framework for uav-ugv collaborative route planning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13687–13694. IEEE, 2024.

## Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- RLAssistedATeamsCodeforsharing.zip