

**Developing Computationally Efficient Optimization Framework for
Collaborative Routing of UAV-UGV System**

BY

SUBRAMANIAN RAMASAMY
B.E. in Mechanical Engineering, Anna University, 2019

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical and Industrial Engineering
in the Graduate College of the
University of Illinois at Chicago, 2025

Chicago, Illinois

Defense Committee:

Pranav A. Bhounsule, Chair and Advisor
James D. Humann, Army Research Laboratory
Sivakumar Rathinam, Mechanical Engineering, Texas A&M University
Bo Zou, Civil and Materials Engineering, UIC
Selvaprabhu Nadarajah, Information and Decision Sciences, UIC

Copyright by

SUBRAMANIAN RAMASAMY

2025

ஆள்வினையும் ஆன்ற அறிவும் எனஇரண்டின்
நீள்வினையால் நீஞும் குடி.

To my father Ramaswamy, who has been supportive throughout my life to allow me to pursue my dreams.

To my mother Lavanya, whose encouragement, kindness, and care have inspired me to strive tirelessly toward achieving my goals.

To my family, friends and well-wishers, who are with me through thick and thin, and help me to achieve my goals.

ACKNOWLEDGMENTS

I would like to extend my heartfelt gratitude to everyone who has been a part of my doctoral journey.

First and foremost, I would like to thank my advisor, Dr. Pranav A. Bhounsule, who entrusted me with this project. He played a pivotal role in inspiring me to pursue a PhD, and I am truly grateful for the journey thus far and the opportunity to work on this fascinating and exciting project. His wisdom, guidance, and unwavering belief have left a lasting impact on me as a researcher. I will carry his mentorship with me as invaluable lessons throughout my career.

To my esteemed committee members, Dr. James D. Humann, Prof. Selvaprabhu Nadarajah, Prof. Sivakumar Rathinam, and Prof. Bo Zou, I express my deep gratitude. Your guidance and feedback throughout my thesis journey has been very helpful.

I would also like to extend my sincere gratitude to Dr. James D. Humann from the Army Research Laboratory for his constant support and valuable feedback throughout my work. Your assistance has been greatly appreciated.

I would like to express my thanks to Dr. Mamadou Seck and Dr. Keivan Ghoseiri from Amtrak. Though the work I did at Amtrak is not directly related to my PhD thesis, their mentorship and guidance during my internship has helped me to gain some insights about how to address research problems that involve non-technical challenges aside technical ones during this PhD journey.

ACKNOWLEDGMENTS (Continued)

To my past and present lab mates, my companions on this academic adventure, you have been a source of great camaraderie and friendship. From Dr. Ernesto Hernandez, Prashanth and Abhishek to my current lab mates Salvador, Jim, Daniel, Safwan, Chun-Ming and Ragib its been an honor to walk this path with you.

Finally, I extend my deepest gratitude to my familythe foundation of my lifeincluding my parents, friends, and loved ones. Your unwavering support and belief in me have brought me this far. Your companionship and the joyful moments we shared kept my sanity in check and helped me pass through some challenging times. I owe every success to your encouragement and presence. I thank you all from the bottom of my heart.

SR

PREFACE

This project was funded by the Army Research Laboratory (ARL) grant.

SUBRAMANIAN RAMASAMY
April 4, 2025

CONTRIBUTION OF AUTHORS

Chapter 2: A significant portion of this chapter is reproduced from a published paper with the following citation: [Ramasamy, S., Reddinger, J. P. F., Dotterweich, J. M., Childers, M. A., & Bhounsule, P. A. (2022). 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.] in which Pranav A. Bhounsule, Subramanian Ramasamy, Jean-Paul F. Reddinger, James Dotterweich and Marshal A. Childers equally contributed to conceiving the idea; Subramanian Ramasamy carried out the data collection, experimental simulations and wrote the manuscript; Pranav A. Bhounsule and Jean-Paul F. Reddinger advised and edited it.

Chapter 3: A significant portion of this chapter is reproduced from a published paper with the following citation: [Ramasamy, S., Mondal, M. S., Reddinger, J. P. F., Dotterweich, J. M., Humann, J. D., Childers, M. A., & Bhounsule, P. A. (2022, June). Heterogenous vehicle routing: comparing parameter tuning using genetic algorithm and bayesian optimization. In 2022 International Conference on Unmanned Aircraft Systems (ICUAS) (pp. 104-113). IEEE.] in which Pranav A. Bhounsule, Subramanian Ramasamy, Md Safwan Mondal, Jean-Paul F. Reddinger, James D. Humann, James Dotterweich and Marshal A. Childers equally contributed to conceiving the idea; Subramanian Ramasamy and Md Safwan Mondal equally contributed and carried out the data collection, experimental simulations and wrote the manuscript; Pranav A. Bhounsule, Jean-Paul F. Reddinger and James D. Humann advised and edited it.

CONTRIBUTION OF AUTHORS (Continued)

Chapter 4: A significant portion of this chapter is reproduced from a published paper with the following citation: [Ramasamy, S., Mondal, M. S., Reddinger, J. P. F., Dotterweich, J. M., Humann, J. D., Childers, M. A., & Bhounsule, P. A. (2023, June). 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) (pp. 95-102). IEEE.] in which Pranav A. Bhounsule, Subramanian Ramasamy, Md Safwan Mondal, James D. Humann, Jean-Paul F. Reddinger, James Dotterweich and Marshal A. Childers equally contributed to conceiving the idea; Subramanian Ramasamy carried out the data collection, experimental simulations and wrote the manuscript; Pranav A. Bhounsule, Md Safwan Mondal and James D. Humann advised and edited it.

Chapter 5: A significant portion of this chapter is reproduced from a published paper with the following citation: [Ramasamy, S., Mondal, M. S., Humann, J. D., Dotterweich, J. M., Reddinger, J. P. F., Childers, M. A., & Bhounsule, P. A. (2024, August). 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) (pp. 204-211). IEEE.] in which Pranav A. Bhounsule, Subramanian Ramasamy, Md Safwan Mondal, James D. Humann, James Dotterweich, Jean-Paul F. Reddinger, and Marshal A. Childers equally contributed to conceiving the idea; Subramanian Ramasamy carried out the data collection, experimental simulations and wrote the manuscript; Pranav A. Bhounsule, Md Safwan Mondal and James D. Humann advised and edited it.

TABLE OF CONTENTS

<u>CHAPTER</u>		<u>PAGE</u>
1 INTRODUCTION		1
1 Expected significance		2
2 Contributions		4
2.1 A Bi-level Optimization Algorithm for Solving Collaborative Heterogeneous Routing Problem		4
2.2 A-Teams: An Optimization Framework to Solve for UGV-UAV Routing		5
2.3 Computationally Efficient Framework: Proposed Variant of A-Teams and its practical applicability		6
2.4 RL-Assisted A-Teams for Adaptive Algorithm Selection in UGV-UAV Optimization		7
2 BACKGROUND		8
1 Vehicle Routing Problem		8
2 Vehicle Routing Problem for Unmanned Aerial Vehicles		9
3 Multiple Unmanned Aerial Vehicle Routing Problem		10
4 Multiple Unmanned Aerial Vehicle Routing Problem with Unmanned Ground Vehicle as mobile recharging vehicle		11
5 Solving cooperative UAV-UGV routing as bi-level optimization problem		13
5.1 Formulating and solving UGV Routing		14
5.2 Formulating UAV Routing as Energy-Constrained Vehicle Routing Problem (E-VRP) and solving using Exact Methods		14
3 HETEROGENEOUS VRP: BI-LEVEL OPTIMIZATION BY PERFORMING GENETIC ALGORITHM TUNING FOR UGV ROUTING AND GRAPH LOCAL SEARCH FOR UAV ROUTING		24
1 INTRODUCTION		25
2 METHODS		30
2.1 Problem statement		30
2.2 Solution approach		31
2.3 Heuristics for UGV (Outer-loop)		31
2.4 Optimizing UAV route (Inner-loop)		33
2.5 Solution using Constraint Programming (CP)		39
2.6 Optimizing the parameters of the UGV heuristics		42
2.6.1 Genetic Algorithm		43
2.6.2 Bayesian Optimization		45

TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>		<u>PAGE</u>
3	RESULTS	47
4	DISCUSSION	50
5	CONCLUSIONS AND FUTURE WORK	52
4	SOLVING VEHICLE ROUTING PROBLEM FOR UNMANNED HETEROGENEOUS VEHICLE SYSTEMS USING ASYNCHRONOUS MULTI-AGENT ARCHITECTURE (A-TEAMS)	55
1	INTRODUCTION	56
2	METHODS	61
2.1	Conventional Two-level optimization	62
2.2	Description of the proposed architecture - A-Teams	63
2.2.1	Nelder-Mead algorithm	65
2.3	Heuristics for UGV (Outer-level)	68
2.4	Optimizing UAV route (Inner-level)	68
3	RESULTS	69
4	DISCUSSION	74
5	CONCLUSIONS AND FUTURE WORK	77
5	COMPUTATIONALLY EFFICIENT MULTI-AGENT OPTIMIZATION FRAMEWORK FOR ONLINE ROUTING OF UAV-UGV SYSTEM: VARIANT OF A-TEAMS	81
1	INTRODUCTION	82
2	RELATED WORK	83
3	METHODS	86
3.1	Problem formulation	86
3.2	Proposed optimization framework	91
3.2.1	Solving outer-level UGV routing using Asynchronous Teams (A-Teams) framework	91
3.2.2	Solving E-VRP for inner-level UAV routing	92
3.3	Hardware setup	94
4	RESULTS	96
4.1	Evaluation of the proposed framework	96
4.2	Hardware re-planning with dynamic changes	101
5	DISCUSSION	104
6	CONCLUSION	106
6	REINFORCEMENT LEARNING ASSISTED A-TEAMS FOR ADAPTIVE ALGORITHM SELECTION IN UAV-UGV OPTIMIZATION PROCESS	108
1	INTRODUCTION	109
2	RELATED WORK	109
3	METHODS	116

TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>		<u>PAGE</u>
3.1	Problem statement	116
3.2	Method 1: GA at outer-level and Constraint Programming (CP) at inner-level	121
3.3	Method 2: Conventional A-Teams at outer-level and CP at inner-level	122
3.4	A-Teams with Predictor Agent at outer-level and CP at inner-level	123
3.4.1	Solving outer-level UGV routing using Asynchronous Teams (A-Teams) framework with Predictor Agent	123
3.5	Reinforcement Learning framework for High-level Decision Making	124
3.5.1	Policy Objective Function	128
3.5.2	Advantage Estimation	129
3.5.3	Value Function Loss	130
3.5.4	Total Objective Function	130
3.5.5	Neural Network Architecture	130
3.5.6	Solving E-VRP for inner-level UAV routing	132
4	RESULTS	134
4.1	Study 1: Comparison of different methods across various task point distributions with 1 UAV and 1 UGV routing	140
4.2	Study 2: Comparison of different methods across various task point distributions with multi-UAV and 1 UGV routing	142
4.3	Study 3: Case Study - Bridge Inspection Using Autonomous Vehicle System	144
5	DISCUSSION	153
6	CONCLUSION	158
7	SOFTWARES USED	160
1	Optimization method	160
8	CONCLUSION	161
	APPENDICES	165
	Appendix A	166
	Appendix B	168
	Appendix C	170
	Appendix D	172
	CITED LITERATURE	174
	VITA	184

LIST OF TABLES

<u>TABLE</u>		<u>PAGE</u>
I	Constraint Quantity analysis	22
II	Parameter set and their range for GA/BO optimization	48
III	Optimal parameters after GA/BO optimization	49
IV	Comparison between GA/BO on metrics from the optimal solution.	54
V	UGV parameters and their ranges (outer loop)	72
VI	Comparison of total cost between A-Teams and conventional two-level optimization for different scenarios	74
VII	Comparison between A-Teams and conventional two-level optimization on metrics from the optimal solution for different scenarios. These results are shown for a specific initialization of population by Constructor agent.	80
VIII	Optimized solution and computational comparison between proposed framework and standard meta-heuristics	98
IX	Predictor agent: Assessment of classification performance with proposed Ensemble vs simpler k-NN model	99
X	Comparison across different optimization methods for high-dense task point distribution with 1 UAV-1 UGV routing.	138
XI	Comparison across different optimization methods for moderate-dense task point distribution with 1 UAV-1 UGV routing.	139
XII	Comparison across different optimization methods for sparse-dense task point distribution with 1 UAV-1 UGV routing.	139
XIII	Comparison across different optimization methods for high-dense task point distribution with 2 UAV-1 UGV routing.	142
XIV	Comparison across different optimization methods for moderate-dense task point distribution with 2 UAV-1 UGV routing.	143
XV	Comparison across different optimization methods for sparse-dense task point distribution with 2 UAV-1 UGV routing.	143
XVI	Average value of Average Age period across all nodes across different vehicle system vs different starting point.	151

LIST OF FIGURES

<u>FIGURE</u>		<u>PAGE</u>
1	Scenario description from Maini et. al [1] for Cooperative UAV-UGV routing	12
2	Different instances of scattered mission point scenarios with $k = 3$ clusters	16
3	An example scenario: (a) The mission scenario. Both, the UAV and UGV start from the same starting location. The range of the UAV is shown using a blue circle. (b) (c) Two possible solutions for mission coverage.	27
4	The problem scenario. The UAV and UGV, both start from the recharging depot. The mission points are shown with black dots. The UAV range is shown with a blue circle.	28
5	Overview of the algorithm	30
6	Heuristics for UGV route. The UGV can make two stops on any mission points such that the first one inside the red ellipse shown with dashed lines and the second is inside the blue ellipse shown with dash-dotted lines. At each stop, the UGV can choose a time to wait (e.g., to recharge the UAV).	32
7	Move operators using in Constraint Programming [2]	40
8	Solution produced by Genetic Algorithm. The plots show the UAV and UGV route at different time spans.	50
9	Solution produced by Bayesian Optimization. The plots show the UAV and UGV route at different time spans.	51
10	Overview of the bi-level optimization algorithm. The outer-level block is run in parallel on multiple cores.	61
11	Implementation of A-Teams architecture for this cooperative routing problem.	62

LIST OF FIGURES (Continued)

<u>FIGURE</u>		<u>PAGE</u>
12	Graphical representation of the Nelder-Mead algorithm for 2-dimensional input example. a) The example inputs under consideration. b) Reflection operation. This operation is performed to make the worst vertex (x_3) of the simplex shape at least better than the second worst vertex (x_2). The reflection is performed to mirror x_3 about the centroid of the simplex shape, and the reflected point is named x_r . The x_r is found via the equation in line 8 of the algorithm Algorithm 3. Based upon the objective function evaluation at x_r , either of the following operations <i>expansion</i> , <i>contraction</i> or <i>shrink</i> is performed accordingly. c) Expansion operation. This operation is performed if the reflected solution x_r is not only better than x_2 , but also better than the current best solution x_1 . If this is the case, then the solution x_r is updated into x_e by moving along that reflected vector direction, and now the new worst solution point will be x_2 . The x_e is found via the equation in line 10 of the algorithm Algorithm 3. d) Contraction operation. This operation is performed if the reflected solution x_r did not get better than x_2 solution. There are two types of contraction. Outside contraction x_{oc} is performed at a third-quarter of the distance between x_3 and x_r and Inside contraction x_{ic} is performed at a quarter of the distance between x_3 and x_r . Comparing these two contractions, x_3 will get updated to x_{ic} or x_{oc} by picking the better solution between them. The x_{oc} or x_{ic} is found via the equation in line 12 or line 14 of the algorithm Algorithm 3. e) Shrink operation. If none of the above operations improve the solution x_3 , then this operation is performed where it updates the current vertices x_2 and x_3 to a location as per the equation in line 15 of the algorithm Algorithm 3.	67
13	Description of different scenarios. The UAV and UGV, both start from one of the recharging depots. The task locations are shown with black dots. The UAV range is shown with a blue circle. a) Scenario 1 b) Scenario 2 c) Scenario 3.	69
14	Description of different scenarios with parameters to be optimized. a) Scenario 1 b) Scenario 2 c) Scenario 3.	73
15	Solution produced by conventional two-level optimization and A-teams on Scenario 1 are indistinguishable and are shown here. The different plot shows the UAV and UGV route at various time-steps.	75
16	Optimal parameter results of respective scenarios obtained using the A-Teams architecture. (a) Scenario 1 (b) Scenario 2 (c) Scenario 3 . . .	76

LIST OF FIGURES (Continued)

<u>FIGURE</u>	<u>PAGE</u>
17 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 frame rate. a) A candidate UGV-UAV task visit points. 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.	83
18 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	87
19 Hardware experimental framework	95
20 Scenario descriptions containing the UGV free route parameters for the optimization process. Primary rendezvous location parameter set (orange and blue ellipse) are solved from Minimum Set Cover problem a) Scenario 1 b) Scenario 2 c) Scenario 3	96
21 Comparison of the Predictions made by Ensemble model vs kNN model for Scenario 2. Left - Ensemble Model; Right - kNN Model	100
22 2D plot of experimental scenario	100
23 Experiment instance with dynamic changes	102
24 UAV-UGV simulated route plan obtained from the multi-agent optimization framework (A-Teams). Top Row shows the initial planned route without any dynamic changes. Bottom Row shows re-planned routes considering the dynamic changes in the scenario. Animation of simulated routes can be viewed here: http://tiny.cc/mgjkxz	103
25 Hardware vs simulation Time-of-Flight comparison of lab setup experiment	105
26 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 frame rate. a) A candidate UGV-UAV task visit points. 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.	110
27 Description of the conventional A-Teams optimization framework used for UGV-UAV routing. This A-Teams inspired from the works of Talukdar et. al 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.	117

LIST OF FIGURES (Continued)

<u>FIGURE</u>		<u>PAGE</u>
28	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	117
29	Markov Decision Process in this study	126
30	Action space for DRL based approach	127
31	RL Training with different Learning Rates	135
32	An instance of training scenario	136
33	Example of different task point distribution scenario types. a) High Dense scenario b) Moderate Dense scenario c) Sparse Dense scenario.	138
34	Computational time comparison across different methods: For 1 UAV - 1 UGV study	140
35	Route sequence of Scenario 1 of High Dense distribution with 1 UAV - 1 UGV routing	141
36	Computational time comparison across different methods: For 2 UAV - 1 UGV study	144
37	Route sequence of Scenario 1 of High Dense distribution with 2 UAV - 1 UGV routing	145
38	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.	147
39	Average age period heatmap. a) For 1 UAV - 1 UGV system. b) For 2 UAV - 1 UGV system.	149
40	Case Study scenario with different starting points. The Green box in (a) and (b) represents the location from where the UAV-UGV inspection starts.	151
41	Case Study scenario routing sequence. Black stars on the right plot represent the dynamic appearance of inspection points to be visited.	152
42	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 Figure 41).	154
43	RL Policy-Driven Optimization Sequence for Agent Selection	155
44	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.	157

LIST OF ALGORITHMS

<u>ALGORITHM</u>	<u>PAGE</u>
1 Genetic Algorithm	43
2 Bayesian Optimization	47
3 Nelder-Mead Method	78
4 A-Teams Architecture	79
5 PPO Algorithm	131

LIST OF ABBREVIATIONS

UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VRP	Vehicle Routing Problem
E-VRP	Energy-Constrained Vehicle Routing Problem
A-Teams	Asynchronous Teams
MIP	Mixed Integer Programming
MILP	Mixed Integer Linear Programming
RAAT	RL-Assisted A-Teams

SUMMARY

Unmanned Aerial Vehicles (UAVs) are increasingly popular for autonomous tasks such as surveillance, package delivery, and disaster relief. However, as these applications expand, so do the challenges of enabling UAVs to perform these tasks over extended durations or on a larger scale. One solution is to pair UAVs with Unmanned Ground Vehicles (UGVs), which can serve as carriers to transport UAVs or as mobile recharging stations to extend their operational range.

To achieve effective performance and coordination between UAVs and UGVs, optimal route planning is essential. The UAV-UGV routing problem, being an NP-Hard combinatorial optimization challenge, is further complicated by the heterogeneity of these vehicles. Traditional optimization approaches, including exact methods like Branch and Bound with MILP formulations, as well as standard heuristics and metaheuristics, are often too time-consuming to be practical due to the dynamic nature of the environment and the autonomous vehicles involved. Thus, a faster, more adaptive planning method is crucial to address these challenges effectively.

This thesis focuses on solving this collaborative UAV-UGV routing problem in a computationally efficient manner such that the solutions are obtained quick enough to be feasibly applied on to the actual UAV-UGV system in a realistic environment. The core concept of solving such a collaborative routing problem is by formulating them as bi-level optimization problem, where the outer-level handles the UGV optimization and for each UGV candidate route, the inner-level handles the UAV optimization. The computational efficiency aspect is

SUMMARY (Continued)

made possible by developing a multi-agent optimization framework, where the agents consist of different optimization algorithms integrated into it. Those algorithms are complementary to each other such that the optimization happens to the best of its ability by picking the best performance out of the algorithms used.

Initial studies concentrated on leveraging the existing multi-agent optimization framework known as A-Teams, incorporating Nelder-Mead and Genetic Algorithms with complementary features for outer-level UGV optimization. The framework was tested across various instances to evaluate its performance compared to using only Genetic Algorithm at the outer level. Each of these approaches employs local search heuristics at the inner level for UAV optimization.

Subsequently, the existing A-Teams framework is developed to be more smarter by integrating a newer 'Predictor agent' into the framework, which imparts slightly better computational efficiency compared to the original A-Teams framework. This A-Teams variant is novel in the sense of adding the Predictor agent into the framework.

Finally, an end-to-end autonomous approach is developed to enhance computational efficiency by enabling decision-making during the optimization process. This approach involves creating a learning-based hyper-heuristic method using Reinforcement Learning (RL) within the multi-agent framework to deliver computationally efficient, real-time UAV-UGV routing solutions. This approach increases the autonomy of the framework, requiring minimal user input.

CHAPTER 1

INTRODUCTION

The world is currently experiencing significant technological advancements, driven by the increasing availability of computational resources. With computing power now accessible at the palm of our hands, applications that were once considered too complex or difficult to implement are becoming feasible and increasingly integrated into everyday life. One such realization is the rapid growth of the usage of Unmanned Aerial Vehicles (UAVs), which are popularly known as Drones or Quadcopters because of its rapid on-board computing capabilities. Their growing popularity is largely due to their speed, agility, and compact design, which allows them to navigate through intricate areas and undertake tasks that are hazardous or challenging for humans. This makes these aerial vehicles ideally suitable for applications such as surveillance, reconnaissance, package delivery, environment and traffic monitoring, search and rescue amongst a plethora of other practical usages. However, a key bottleneck of using such UAVs is their limited battery capacity, which typically restricts them to about 15–20 minutes of flight time [1]. With this limitation in the flight time, it becomes difficult to exploit the drones in above applications where it demands either large scale task operations or longer durations. However, there are solutions to overcome this problem. For instance, the flight time and subsequently the coverage area of UAVs can be increased by having several recharging depots spread across the area. But, in non-urban environments such as rural areas or hostile environments, it would be expensive or tactically impossible to have recharging depots placed in advance [2]. Hence,

a viable alternative is to have UAVs recharge on Unmanned Ground Vehicles (UGVs). These UGVs can not only work as a team with UAVs in performing tasks, but also can provide UAV recharging. That is, when the UAV is low on its battery, it coordinates with the UGV to find a rendezvous point, lands on the UGV, get recharged and then continue to perform tasks.

Solving this problem could be computationally challenging as it necessitates spatial and temporal coordination between these two heterogeneous vehicles. UAV is fast and power hungry; whereas UGV is slow but resourceful. In order to utilize this team to perform tasks as efficient as possible, an optimization problem has to be solved where the formulation depends upon the application. In this case, the application is taken to be a surveillance problem where given a set of task points to visit, number of UAVs and UGVs, the problem is to plan an optimal route for the UAV-UGV team such that the tasks are performed efficiently and UAVs get recharged in a timely manner to perform such tasks thereby they never run out of charge. Hence, this problem falls within the category of advanced Vehicle Routing Problems, a type of combinatorial optimization problem, where the objective is to minimize an objective function such as cost, time or energy consumption of the robots while satisfying a set of constraints in their visits. Hence, additionally this problem falls under the NP-Hard problem category, adding further complexity. The goal of this thesis is to investigate different possible strategies and develop algorithms that achieves efficient computation to solve such combinatorial optimization problem.

1 Expected significance

Presently, there is an increasing demand in applications of drones for a wide variety of tasks ranging from military to civilian applications (put some references for this). Because of this,

their operational capabilities should be on par to satisfy the demand of a certain application. These drones are popular, but they are limited in the duration of tasks that they can perform. In remote areas, where setting up fixed recharging depots is not practically feasible, one of the potential alternatives is to team it up with other autonomous vehicles such as UGV to expand their usage for critical tasks like surveillance and disaster relief. However, using multiple heterogeneous vehicles involves an additional layer of complexity. This is because, apart from planning an optimal path for these respective mobile robots in order to perform certain tasks, they also have to be co-operated optimally so that the overall cost is minimized, opening up an advanced category of Vehicle Routing Problems known as the Cooperative Vehicle Routing Problem. Solving these problems are NP-Hard, and hence in order to achieve practicality of using such heterogeneous mobile robots for routing, an optimization approach has been developed so that it has the capability to solve for optimal solution very quickly so that these methods can be used on board the robots in real-time. Also, this approach has the capability to adapt its solution to encounter any dynamic changes in the environment/scenario considered. Beyond vehicle routing problem, this approach holds promise to be applied to any kind of combinatorial optimization problems given the right components as per the application to be considered.

2 Contributions

2.1 A Bi-level Optimization Algorithm for Solving Collaborative Heterogeneous Routing Problem

The first contribution of this thesis lies in developing a novel bi-level optimization method for solving such co-operative vehicle routing problem. This is because, since the UAV gets recharged on the UGV, they have this interdependency in their routes where UAVs and UGVs need to get coordinated make rendezvous at certain points in order for the UAV to get recharged. Throughout this thesis, 'UGV first, UAV second' method will be followed. Thus, the outer-level of the bi-level optimization method constitutes optimizing the UGV routes, and for a corresponding UGV route, the inner-level constitutes optimizing the UAV routes for that UGV route. Post UAV-optimization, the combined UAV-UGV route is obtained, which is given as a feedback to perform UGV optimization. Thus, this bi-level optimization method will be a closed-loop optimization where the solution quality of the UAV-UGV route will be utilized to perform the UGV optimization. In this project, heuristic algorithms are utilized to perform the route optimization at both UGV and UAV levels.

Alternative approaches involve implementing exact solution methods like Mixed Integer Linear Programming (MILP) to solve for such cooperative routing at both UGV and UAV levels, but such methods often become intractable with increase in the problem size (cite our journal paper). This project introduces a novel approach to formulate the UGV route as a set of free route parameters to be optimized. This enables us to reduce the computational burden

of graph search method at the outer-level optimization so that the graph search is just utilized for UAV routing. These contributions are explored in-depth in Chapter 3.

2.2 A-Teams: An Optimization Framework to Solve for UGV-UAV Routing

The second key contribution of this thesis is to build upon the simpler bi-level optimization concept to develop a framework that efficiently computes the optimal solution for the coordinated vehicle path planning by utilizing several algorithms as optimizing agents concurrently to produce an optimal solution faster without compromising on its quality. Although previous approach of implementing Genetic Algorithm for UGV route optimization help us to achieve globally optimal solutions, they come at a cost of significant computational time. This is because, such algorithms perform search over a large space, compromising on their efficiency. Hence they can be used when the computational time is not critical. But in this problem of performing route planning for autonomous mobile robots, time is critical to perform realistic experiments. In contrast to that, local optimization algorithms provide quicker solutions, but are optimal in a small region of space.

This project introduces a novel approach for solving a bi-level UGV-UAV optimization problem where the UGV routing is performed using a multi-agent optimization framework called A-Teams (Asynchronous Teams). The optimization agents in the A-Teams framework is proposed by Sachdev [] where these agents worked together cooperatively to perform optimization on a given problem. There are three main agents that constitute this framework: Constructor Agent, Improver Agent and Destroyer Agent. So far the A-Teams in the literature has been limited to solving basic routing and other combinatorial optimization problems. This novel ap-

proach involves solving a heterogeneous vehicle routing problem with implementing A-Teams at the outer-level UGV routing while the UAV routing at the inner level is solved by formulating it as Vehicle Routing Problem with charge, time and optional node visit constraints.

Additionally, this novel approach introduces the application of concept of hybrid algorithms for solving UGV route parameter optimization. Since global optimization algorithm searches over the entire search space, while local optimization algorithms searches locally, this approach uses a combination of local and global optimization algorithms as a part of the A-Teams. Thus, using these hybrid algorithms that are complementary in nature together can solve the optimization problem more efficiently. This is because, while the global optimizer searches over the entire UGV route parameter search space, the local optimizer simultaneously tries to improve the current best candidate UGV route locally thereby the number of overall searches can be reduced. These contributions are explored in-depth in Chapter 4.

2.3 Computationally Efficient Framework: Proposed Variant of A-Teams and its practical applicability

So far, the novelty was utilizing the existing A-Teams to perform bi-level optimization for cooperative UAV-UGV routing. The A-Teams in the existing literature revolves around three main agents such as Constructor, Improver and Destroyer Agents.

The third key contribution of this thesis to impart novelty to the existing A-Teams. In this approach, a newer agent had been integrated into the framework called 'Predictor Agent'. The purpose of this agent is to predict the quality of a particular UGV candidate route parameter before even it gets passed onto UAV optimization. The prediction is made by a trained Machine

Learning model that performs supervised learning of mapping several UGV route parameters to corresponding UGV-UAV solutions. Integrating this agent constitutes the fusion of Machine Learning into the conventional optimization methods. Thus, the combination of these four different agents in the proposed variant of A-Teams cooperatively yields optimal solutions more efficiently without compromising on its optimality.

Additionally, this novel approach enables the overall UGV-UAV optimization to be implemented in practical settings, where the framework extends its potential to perform re-planning when scenario conditions change. The frameworks capability is validated with an hardware experiment and these contributions are explained in-depth in Chapter 5.

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

The fourth key contribution of this thesis lies in automating the proposed A-Teams variant. The A-Teams framework comprises multiple agents, each with user-defined parameters. This approach leverages a Reinforcement Learning (RL) policy as a decision-making system to determine which agents should operate at each step of the optimization process. The RL policy selectively activates the agents that most effectively contribute to the optimization, thereby reducing computational time without compromising the quality of the UAV-UGV end solution. By intelligently managing agent participation, the automated A-Teams framework enhances both efficiency and performance in achieving optimal routing solutions. The novel framework is put to test across different cases including multi-UAV routing and dynamic scenario changes. These contributions are detailed in Chapter 6.

CHAPTER 2

BACKGROUND

1 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a type of combinatorial optimization problem, that uses integer programming method to solve for the optimal set of routes that a fleet of vehicles traverse in order to deliver or visit a set of customers. That is, VRP seeks to determine the optimal routes for each vehicle to minimize total operational costs, which can include factors such as distance traveled, time spent, and fuel consumption, while adhering to various practical constraints. These constraints can encompass vehicle capacity limits, delivery time windows, route length restrictions, and specific service requirements. Solving the VRP efficiently is critical for optimizing supply chain operations, reducing transportation costs, enhancing customer satisfaction, and minimizing environmental impact. Given its NP-hard nature, finding exact solutions is computationally intensive for large instances. Consequently, a variety of heuristic and metaheuristic algorithms have been developed to provide near-optimal solutions within reasonable time frames.

Parts of this chapter is taken from the following published journal article:
Ramasamy, S., Reddinger, J. P. F., Dotterweich, J. M., Childers, M. A., & Bhounsule, P. A. (2022). 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.

Traditionally, Vehicle Routing Problems (VRPs) have been prevalent in the logistics domain. In these scenarios, routing for a considerable number of trucks is formulated and optimized to execute 1-day or 2-day routes while satisfying a set of constraints. These constraints typically include truckload capacity limitations, as well as requirements for the pickup and delivery of goods from one location to another. Khuller et al. [3] were one of the first ones to solve vehicle routing problem with fuel constraints. They considered the problem of finding the cheapest route for a fuel constrained vehicle with a set of fueling stations, each with a different fuel price. They used a dynamic programming (DP) formulation to solve the problem.

2 Vehicle Routing Problem for Unmanned Aerial Vehicles

With the advent of the latest technologies and advancements in automation, along with the rapid rise of Unmanned Aerial Vehicles (UAVs), new possibilities have emerged for optimizing routing strategies. UAVs offer the capability to visit a set of nodes or waypoints in environments that are hazardous for manual vehicles to traverse. By leveraging UAVs to perform these visits automatically, it is possible to enhance efficiency and safety in logistics operations.

Consequently, the application of UAVs to solve routing problems remains a hot topic for research and exploration. The goal is to achieve (nearly) optimal solutions that are realistically possible, addressing the unique challenges and opportunities they present. Several works have been done in the literature regarding the implementation of the Vehicle Routing formulation for UAV routing. For instance, Kannon et al. [4] considered the problem of finding the route of a single fuel-constrained aircraft to visit a set of waypoints with a set of aerial recharging

way-points. They compared a mixed-integer linear programming (MILP) formulation with a DP formulation and found that DP outperforms MILP.

3 Multiple Unmanned Aerial Vehicle Routing Problem

While UAVs are fast, agile, and compactmaking them suitable for a wide range of applicationsrouting a single UAV is often not practically feasible, unlike typical vehicle routing for a logistic vehicle. This limitation arises because UAVs are power-hungry and their battery capacity is constrained by payload requirements. To overcome these challenges, one effective approach is to employ a swarm of drones that collaborate to perform tasks cooperatively, thereby achieving practical and scalable solutions. This strategy necessitates the implementation of the Vehicle Routing Problem (VRP) for multiple UAVs. To explore this approach, numerous studies in the literature have focused on optimizing the routes of multi-UAV systems, aiming to enhance feasibility in various applications. Some of the works are as follows. Levy et al. [5] and Sundar et al. [6] considered extensions to multiple fuel-constrained Unmanned Aerial Systems (sUASs). The goal here was to minimize the distance traveled by multiple fuel-constrained sUASs to visit a set of waypoints once and recharge on ground-based recharging depots. Levy et al. used a variable neighborhood search based on randomization and variable neighborhood descent based on the gradient to search for an optimal solution. Sundar et al. formulated several MILP formulations and solved these using an off-the-shelf MILP solvers. Similar to these works, Bung Duk Song et al. [7] considered a multiple heterogeneous sUAS path planning problem in which automatic Logistics Service Stations (LSS) are used for sUAS fuel replenishment. Younghoon Choi et al. [8] implemented a new Coverage Path Planning (CPP) problem for solving the

routes of a fleet of sUASs with energy constraints. The authors' novelty comes in formulating a column generation approach to deal with non-linear energy consumption, where traditional arc-based optimization approaches do not accurately estimate such function. Radzki et al. [9] proposed a robust approach to delivery planning of small Unmanned Aerial Systems (sUAS) for disaster relief missions. The authors formulated an algorithm for planning of the routes of sUASs such a way that it ensures the proper execution of a certain delivery mission irrespective of any change in the weather conditions. Andrew et al. [10] solved a two-tier route optimization problem of homogeneous sUASs and repair vehicles for network exploration and failure repair. Their problem first solves for the optimal route of sUASs to explore the potential failure locations followed by the optimal route of repair vehicles to address the failure regions located by the sUASs. Andres et al. [11] proposed a novel approach for global path optimization of sUASs by combining Traveling Salesman Problem and continuous optimal control formulations where the latter takes the sUAS dynamics and constraints into consideration. Sung et al., addressed the optimal zoning problem of Unmanned Aerial Vehicles using Genetic Algorithm to optimize package delivery services.

4 Multiple Unmanned Aerial Vehicle Routing Problem with Unmanned Ground Vehicle as mobile recharging vehicle

Recent works in literature have started exploring the avenues of coordinating heterogeneous vehicles by implementing aerial-ground vehicle collaboration in order to harness the potential of utilizing such a system for large scale and long duration scenarios. Maini et al. [1] considered the problem of routing a single fuel-constrained sUAS to a set of missions while being recharged

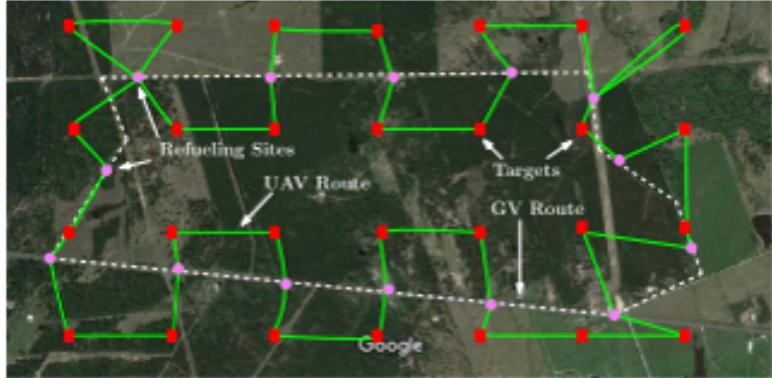


Figure 1. Scenario description from Maini et. al [1] for Cooperative UAV-UGV routing

by stopping at a UGV traveling on a road network. They solved the problem using a two-stage approach. First, using the sUAS range constraints, they found a set of recharging depots. Second, they formulated a mixed-integer linear program and solved for the path of both the sUAS and UGV. Manyam et al., [12] solved the problem of cooperative routing of sUAS-UGV to visit a set of mission points while being within a radius of each other to enable communication. They cast the resulting problem as a mixed integer programming problem and solved using a custom-written branch-and-bound algorithm. Petitprez et al., [13] considered deployment of sUAS and UGV such that the UGVs pick up sUASs that have land on fixed locations after visiting mission points. The objective was to minimize the operational cost while maximizing the inspection performance, a contradictory set of objectives that are solved using multi-objective optimization using genetic algorithms and capacitated arc routing problem. Their problem did not have a temporal aspect to it. Liu et al., [14] considered the problem of choosing recharge stations and flight routes for sUAS to visit given mission points. They cast the problem as

a binary optimization problem, but solved it using heuristics that first cluster the mission points and then use local searches to solve for a path. Bard et al. [15] considered the problem of clustering a set of customers into zones such that a single vehicle can serve all the zones while meeting delivery and pick-up times. Since the pickup and delivery schedule is random, a probabilistic traveling salesman problem is formulated and solved using heuristics that break the problem into two parts, solve them individually, and then connect back to form the complete solution. Dondo et al., [16] used clustering approach to solve a heterogeneous fleet vehicle routing problem with time windows. Initially, the customers are clustered based upon node locations, load, and time windows. Then vehicles are assigned to the cluster followed by their sequencing. Finally, the nodes within the cluster are ordered and arrival times are computed. They used a standard MILP formulation which was solved using branch and bound.

5 Solving cooperative UAV-UGV routing as bi-level optimization problem

The overall objective is to plan the path of fuel-limited Unmanned Air Vehicles (UAV) to pre-specified mission points while minimizing the total distance travelled by all UAV. The UAV may be recharged by docking on Unmanned Ground Vehicles (UGV). The path of the UGV and UAV-UGV rendezvous locations also needs to be computed.

One of the major challenges in this kind of collaborative Unmanned Aerial-Ground Vehicle problems is that the vehicles involved are heterogeneous in nature. That is, UAVs, while they are agile, are power hungry and they need to get recharged from time to time for long range missions. Meanwhile, UGVs have a greater power capacity but they are slow in their travel. Because of this, the problem becomes challenging when UGV can provide power to UAVs when

they are docked. Since the UAVs can get recharged on UGVs, the UAV routes depend upon the UGV route. Because, based upon the way the UGV traverse, the flight range of the UAV needs to be made sure that the UAV can reach the UGV in time. Hence, if the UGV route is solved first, then the UAV route can be solved based upon the UGV route. This necessitates to solve the entire collaborative routing as a bi-level optimization problem.

5.1 Formulating and solving UGV Routing

Since the approach ‘UGV first, UAV second’ is considered to solve such cooperative UAV-UGV routing, the UGV route has to be formulated to solve the UAV routing. The UGV has to visit the points in such a way that those UGV points would help to achieve the complete radial coverage of UAVs. Hence, along the UGV’s travel, they have to wait to accommodate the UAVs at strategic UGV points such that it ensures that the UAV is within the range (or radial coverage area) to facilitate feasible landing, recharging and extend the task visits of the UAVs. So to formulate the UGV routing, strategic placing of UGV points needs to be made and those are fed to UAV optimization as constraints. Initial studies formulated the UGV optimization problem as a Traveling Salesman Problem using an integer linear programming approach. More details about this can be found in [17].

5.2 Formulating UAV Routing as Energy-Constrained Vehicle Routing Problem (E-VRP) and solving using Exact Methods

Now we have a formulation for the UGV, we assign those strategic UGV points on its path as possible recharging depots for the UAV-UGV rendezvous. We assume that each of the K UAVs starts at the same location as the UGV, formulate a vehicle routing problem with capacity

constraints to account for fuel limits, time windows to allow for rendezvous with UGV, and dropped visits to allow the UAVs to visit some of the many vertices on the UGV path.

Initially to put this bi-level optimization with UGV first, UAV second approach to test, a preliminary analysis was made with different kinds of scattered mission points (Figure 2) with simple constraints and assumptions as follows: The velocity of the sUAS is fixed and the velocity of the UGV is bounded. The sUAS battery capacity is fixed, while the UGV has unlimited fuel. Since the velocity is constant, the battery capacity of the sUAS is assumed to be directly proportional to the flight time. The sUAS is assumed to dock on a stationary UGV during recharging and recharge to full battery capacity. Both UGV and sUAS are stationary during recharging of the sUAS. The recharging time of the sUAS is constant and is independent of the remaining battery capacity. The mission points are assumed to be distributed in the form of ' k ' clusters spanned across the area considered. Figure 2 gives an example of different instances with $k = 3$ number of clusters. Similarly, we considered $k = 2$ or $k = 4$ clusters of mission points and performed the UAV-UGV routing to understand the optimization capability better.

As a preliminary study with example toy scenarios, we constrain the UAV to a fixed speed, pre-specify the battery capacity and service time as the UAV lands and waits on the UGV. With this into consideration, the UAV routing is formulated as follows.

The set of all UGV waypoints is denoted by $D = \{0, 1, 2, \dots, m\}$. There are $n - m$ pre-specified mission points which belong to the set $M = \{m + 1, \dots, n\}$. The set of all vertices is then $V = M \cup D = \{0, 1, 2, \dots, m, m + 1, \dots, n\}$. The set of all edges denotes all possible connections between the vertices $E = \{(i, j) | i, j \in V, i \neq j\}$.

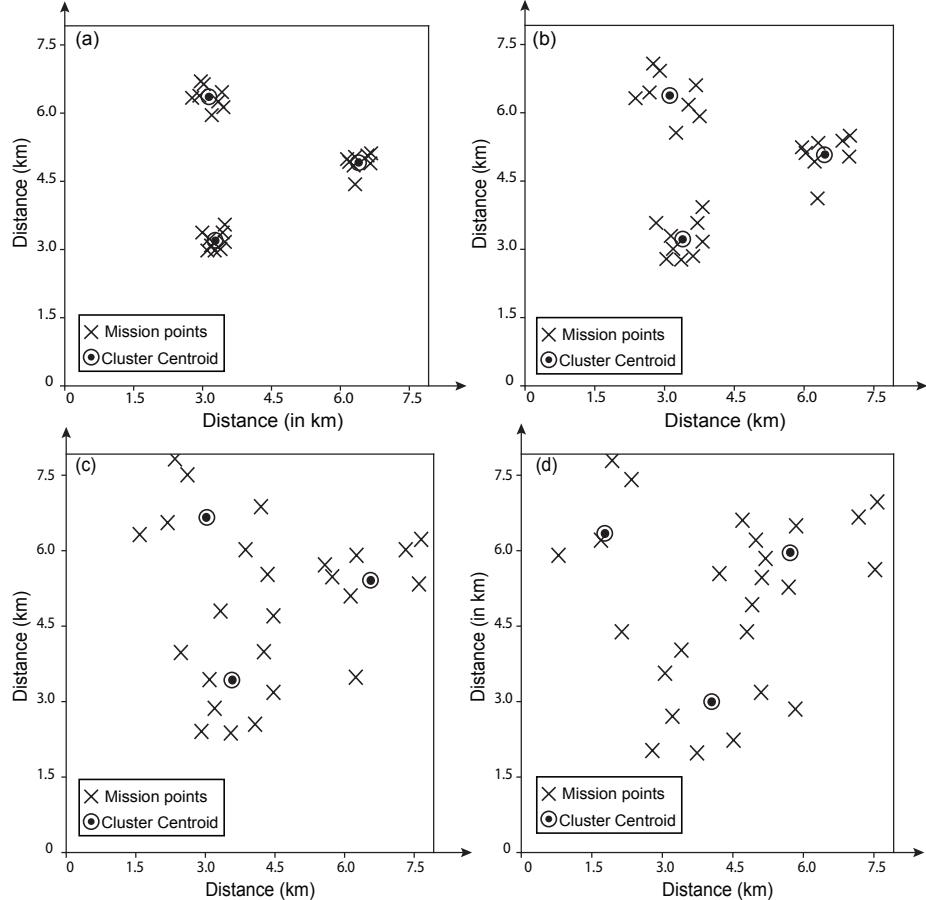


Figure 2. Different instances of scattered mission point scenarios with $k = 3$ clusters

Consider an undirected graph $G = (V, E)$ where V is the entire set of vertices $V = \{0, 1, 2, \dots, m, m + 1, \dots, n\}$ and E is the set of edges that gives the arc costs between i and j and $E = \{(i, j) | i, j \in V, i \neq j\}$. Let c_{ij} be the non-negative arc 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. We formulate the VRP problem with fuel constraints, time windows, and dropped visits as follows,

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \quad (5.1)$$

$$\text{s.t., } \sum_{i \in V} x_{ij} = 1, \quad \forall j \in M \setminus D \quad (5.2)$$

$$\sum_{j \in V} x_{ij} = 1, \quad \forall i \in M \setminus D \quad (5.3)$$

$$\sum_{i \in V} x_{ij} \leq 1, \quad \forall j \in D \setminus \{0\} \quad (5.4)$$

$$\sum_{j \in V} x_{ij} \leq 1, \quad \forall i \in D \setminus \{0\} \quad (5.5)$$

$$\sum_{j \in V} x_{0j} = \sum_{i \in V} x_{im} = K, \quad \{0, m\} \in D \quad (5.6)$$

$$f_j \leq f_i - (c_{ij}x_{ij}) + L_1(1 - x_{ij}), \quad \forall i \in V, j \in V \setminus D \quad (5.7)$$

$$f_j = Q, \quad \forall j \in D \quad (5.8)$$

$$0 \leq f_j \leq Q, \quad \forall j \in V \quad (5.9)$$

$$t_j \geq t_i + (s_i + c_{ij}x_{ij}) - L_2(1 - x_{ij}), \quad \forall i \in V, j \in V \quad (5.10)$$

$$t_j^l \leq t_j \leq t_j^u, \quad \forall j \in V \quad (5.11)$$

$$x_{ij} = 0, \quad \forall i \in D, \forall j \in V \quad (5.12)$$

$$x_{ij} = 1 \rightarrow f_i \geq c_{ij}, \quad \forall i \in V \setminus D, \forall j \in V \quad (5.13)$$

$$x_{ij} = 1 \rightarrow f_i = Q, \quad \forall i \in D, \forall j \in V \setminus D \quad (5.14)$$

$$x_{ij} = 1 \rightarrow \sum_{i \in V \setminus D} x_{ji} = 1, \quad \forall j \in D, \forall i \in V \setminus D \quad (5.15)$$

$$x_{mj} = 0, \quad \forall m \in D, \forall j \in V \quad (5.16)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in V \quad (5.17)$$

$$f_i > 0, \quad f_i \in \mathbb{R}_+, \quad \forall i \in V \quad (5.18)$$

$$t_i > 0, \quad t_i \in \mathbb{Z}, \quad \forall i \in V \quad (5.19)$$

$$s_i \geq 0, \quad s_i \in \mathbb{Z}, \quad \forall i \in V \quad (5.20)$$

$$Q > 0, \quad Q \in \mathbb{R}_+ \quad (5.21)$$

$$L_1, L_2 > 0, \quad L_1, L_2 \in \mathbb{R}_+ \quad (5.22)$$

The objective is Eq. Equation 3.10 is to minimize the total distance traveled by the all the sUASs. Constraints in Eq. Equation 2.3 and Eq. Equation 2.4 represents the flow conservation where the inflow of a certain sUAS should be equal to the outflow of that sUAS at any vertex among the mission vertices M . Constraints in Eq. Equation 2.5 and Eq. Equation 2.6 denotes the optional stops the sUAS can take on the UGV vertices D , i.e., dropped visits. Next, constraint in Eq. Equation 2.7 also represents the flow conservation but here it is represented for start and end vertices, where the number of sUASs leaving the start vertex must be equal to the number of sUASs reaching the end vertex. The start vertex and end vertex correspond to the first and last vertex of the UGV route. The constraint in Eq. Equation 3.11 is the Miller-Tucker-Zemlin (MTZ) formulation [18] for sub-tour elimination. This constraint ensures that none of the sUAS batteries are depleted out while eliminating sub-tours. In this constraint, L_1 denotes a large number. This constraint becomes active only when there is a flow between vertices i

and j and subtracts from the sUAS fuel based on distance between the two vertices. The fuel consumption of sUAS depends upon the distance traveled by them. It is directly proportional to the distance traveled between two vertices i and j . Constraint Eq. Equation 3.12 states that if the vertex is a recharging UGV stop, then UGV has to refuel the sUAS to its full capacity Q . Constraint Eq. Equation 3.13 is the condition that the sUAS's fuel at any vertex in V should be between 0 and maximum fuel capacity. Constraint Eq. Equation 3.14 denotes that the cumulative arrival time at j^{th} node is equal to the sum of cumulative time at the node i , t_i , the service time at the node i , s_i , and the travel time between nodes i and j , $c_{ij}x_{ij}$. Here L_2 denotes a large number which helps to eliminate sub-tour constraints similar to Eq. Equation 3.11. Constraint Eq. Equation 3.15 is the time window constraint that the vehicle visits a certain vertex in the specified time window for that node. In this problem, the mission nodes are not constrained by time as the sUASs have the liberty to visit those mission points that benefits them according to the travel of UGV. This means that whenever sUAS needs to get refueled, it would be easier for it to go to the UGV to refuel. Constraints Eq. Equation 2.13 restricts that the two consecutive visits made by the sUASs should not be consecutive UGV stops. Constraints Eq. Equation 2.14 - Eq. Equation 3.16 represents the indicator constraints where the constraints to the right side of the arrow should hold if the binary decision variable x_{ij} is equal to 1. If x_{ij} is equal to zero, then the constraints to the right side of the arrow may be violated. The constraint in Eq. Equation 2.14 that if there is travel from any mission vertex i to the UGV vertex j , then the fuel level at the i^{th} node should be atleast equal to the distance traveled between them because the fuel consumption in this problem is assumed to be linearly

proportional to the distance traveled. Constraint in Eq. Equation 2.15 indicates that if there is travel from the UGV vertex i to any mission vertex j , then the fuel level at the i^{th} node should be the maximum fuel capacity of the sUAS as it is recharging to its full capacity at the UGV stop. The constraint in Eq. Equation 3.16 makes sure that if any sUAS comes to the refuel vertex to recharge, then there must exist an arc between that refuel node and a mission node to maintain the flow conservation. Constraint in Eq. Equation 2.17 denotes that there should not exist any flow once the vehicle has reached the end node m . Eq. Equation 2.18 is a binary decision variable that is responsible for flow between the edges. Eq. Equation 2.19 represents the continuous decision variable that monitors the fuel level at any node and has zero as the lower bound value. Eq. Equation 2.20 represents the integer decision variable that computes the cumulative time of sUAS's route and has zero as the lower bound. Eq. Equation 2.21 denotes the service time at the respective nodes, which is a positive integer with a lower bound equal to zero. Eq. Equation 2.22 represents the maximum fuel capacity of a sUAS Finally, Eq. Equation 2.23 denotes the large numbers used in the constraints Eq. Equation 3.11 and Eq. Equation 3.14.

Table I gives an itemized lists the number of constraints in the sUAS problem formulation. The equation numbers in the table correspond to the equations from the Sec. 5.2. In the table, the notation V denotes the total number of vertices, including all possible UGV stops and the mission points, while D denotes only the UGV stops. The grand sum of all constraints $SUM = 6V^2 - 2D^2 + DV + 2V - 2$. Thus, the number of constraints scale as the square of the number of mission points.

TABLE I

Constraint		Quantity analysis	
Equation #	Number of constraints	Equation #	Number of constraints
Equation 2.3	$V - D$	Equation 3.14	V^2
Equation 2.4	$V - D$	Equation 3.15	$2 \times V^2$
Equation 2.5	$D - 2$	Equation 2.13	D^2
Equation 2.6	$D - 2$	Equation 2.14	$(V - D) \times D$
Equation 2.7	2	Equation 2.15	$(V - D) \times D$
Equation 3.11	$(V - D) \times V$	Equation 3.16	$((V - D) \times D) - (V - D)$
Equation 3.12	$D \times V$	Equation 2.17	$(V - D)$
Equation 3.13	$(2 \times (V - D)) \times V$		
<hr/> $SUM = 6V^2 - 2D^2 + DV + 2V - 2$ <hr/>			

We utilized the Gurobi Mixed Integer Linear Programming (MILP) optimizer [19], which employs the Branch and Bound algorithm an exact solution method to find the optimal solution. Exact methods are designed to rigorously solve optimization problems, ensuring that the global optimum is obtained. For $k = 2$ clusters, $D = 9$ and $V = 34$ would give a total constraint of 7146 and is solved about 40 sec using Gurobi on a standard desktop (3.7 GHz Intel Core i9 processor with 32 GB RAM on a 64-bit operating system). However, if $k = 4$ then $D = 17$ and $V = 42$ would give constraints of 10802 and is solved in 240 sec using Gurobi. Thus, Gurobi takes significantly higher time as the number of constraints are increased. Thus, it does

not scale very well for a larger number of constraints. Hence, heuristics approaches are via alternatives in this case to balance the tradeoff between optimality and practical applicability.

Since UAVs are autonomous robotic systems, their routing algorithms must inherently possess the ability to swiftly adapt to changes in their surroundings. This adaptability ensures that UAVs can respond in real-time to dynamic environmental conditions, unexpected obstacles, or shifting mission requirements. By striking a balance between both optimality and rapid responsiveness, the algorithm should be devised in a manner such that the routing solutions enhance the practical deployment of UAVs, ensuring they operate efficiently and reliably in ever-changing environments. The following chapters of this thesis will dive deep into researching various heuristic approaches that can be undertaken such that the computational time is drastically reduced without compromising much on the solution optimality.

CHAPTER 3

HETEROGENEOUS VRP: BI-LEVEL OPTIMIZATION BY PERFORMING GENETIC ALGORITHM TUNING FOR UGV ROUTING AND GRAPH LOCAL SEARCH FOR UAV ROUTING

Overview: Heterogeneous vehicles (e.g., unmanned ground vehicle (UGV) and unmanned aerial vehicle (UAV)) are best suited for surveillance application over large areas. UAVs are fast, but fuel limited, while, UGVs have a larger fuel capacity, but are relatively slow. When UAVs are combined with UGVs they can provide larger coverage at a relatively fast speed. The UAV may also be recharged on the UGV as needed. The resulting route optimization problem is computationally complex, but may be solved relatively fast using heuristics. In this paper, we solve for a mission route using a two-level optimization; (1) the UGV route is assigned using heuristics with free parameters, (2) the UAV route is solved using a vehicle routing problem formulation with capacity constraints, time windows, and dropped visits. However, this open-loop two-level optimization may yield non-optimal solutions or fail completely because of poor choice of UGV parameters. Our primary objective is to explore closed loop optimization where the free parameters of the UGV routes are optimized using Bayesian optimization and

Parts of this chapter is taken from the following published journal article:
Ramasamy, S., Mondal, M. S., Reddinger, J. P. F., Dotterweich, J. M., Humann, J. D., Childers, M. A., & Bhounsule, P. A. (2022, June). Heterogenous vehicle routing: comparing parameter tuning using genetic algorithm and bayesian optimization. In 2022 International Conference on Unmanned Aircraft Systems (ICUAS) (pp. 104-113). IEEE.

Genetic algorithms. Our results show that both methods produce good quality solutions, but bayesian optimization is computationally more efficient than genetic algorithm.

Keywords: Write keywords here

1 INTRODUCTION

The availability of simple-to-control and low-cost unmanned aerial vehicles (UAVs) has opened the possibility of practical surveillance systems [20]. UAVs can provide automated surveillance for reconnaissance, weather observations, environment and traffic monitoring, search and rescue, and border patrol [21]. Although UAVs are fast, they are severely limited to relatively small area due their limited battery capacity [22].

To increase their range, UAVs may be combined with unmanned ground vehicles (UGVs). UGVs move slow and are terrain limited, but are large enough to enable docking and recharging of aerial vehicles. They may also be used to survey locations that are accessible through the roads [23]. Thus a system consisting of heterogenous vehicles consisting of UAVs and UGVs are ideally suited for automated surveillance applications (e.g., [24, 25]).

The use of heterogenous vehicles with different capabilities makes the route selection problem more challenging [26]. One has to consider the fuel limitations of the UAV and the speed limitations of the UGV to device successful routing paths. In order to push their limits, optimization of their routes with metrics such as reducing the time and/or the overall fuel consumption are important. In this paper, we provide a framework for optimizing the routes of heterogenous vehicles with resource constraints (e.g., fuel, speed) and terrain constraints (mission spread).

We illustrate the problem using the mission scenario shown in Figure 3 (a). The mission points are shown with black dots. The UAV range for a single charge is shown with a blue circle. The circle indicates that the UAV cannot visit all mission points on a single charge. Figure 3 (b) and (c) shows two possible solutions. In Figure 3 (b), the UGV and UAV move together through the path $a \rightarrow b \rightarrow c$. The UAV then takes off from the UGV and travels the path $f \rightarrow g$ to return to the UGV. Then the UGV and UAV move along $d \rightarrow e$ to return to the starting point. Figure 3 (c) shows an alternate solution. The UGV and UAV move together along a . Then the UAV takes off and its moves along $e \rightarrow f$ to return to the starting point. The UGV moves along $b \rightarrow c \rightarrow d$ and returning to the start point. If optimization criteria is UAV fuel consumption then the strategy in (b) is better because the UAV flies over a shorter distance. If optimization criteria is the total time taken to complete the missions, then (c) is better here because unlike in (b), the UGV does not have to wait for the UAV to land back on it thus (c) takes less time than (b). From this example, we note that the UGV heuristics, the stop location for the UAV to take-off and the wait time are critical based on the optimization criteria. In this study, these parameters are optimized by the genetic algorithm and bayesian optimization.

There has been considerable work on routing of fuel constrained UAVs. Levy et al. [5] and Sundar et al. [27] considered the routing of multiple fuel-constrained Unmanned Aerial Vehicles (UAVs) with recharging on fixed depots. Levy et al. used a variable neighborhood search based on randomization and variable neighborhood descent based on the gradient to search for an optimal solution. Sundar et al. formulated several mixed-integer linear programming (MILP)

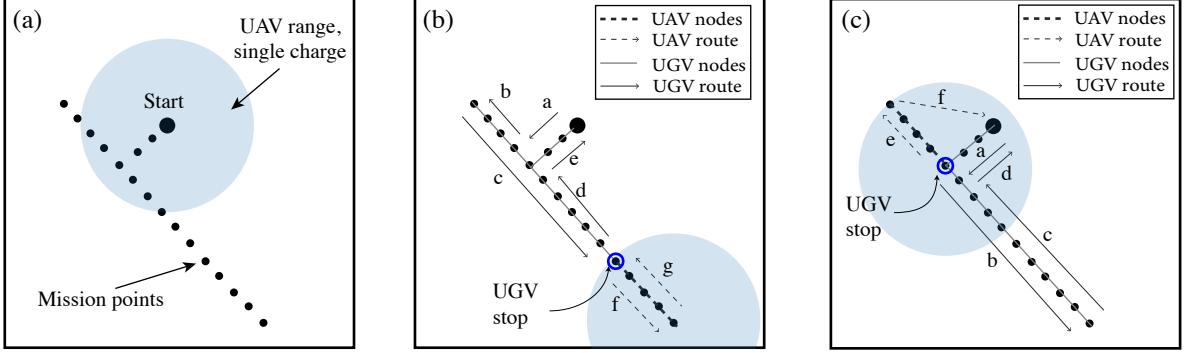


Figure 3. An example scenario: (a) The mission scenario. Both, the UAV and UGV start from the same starting location. The range of the UAV is shown using a blue circle. (b) (c) Two possible solutions for mission coverage.

formulations and solved these using an off-the-shelf MILP solvers. Ren et al. [28] considered a collaborative two-UAV and one-UGV problem where the purpose of UGV as a carrier is to deploy and retrieve the flying robots on time, and the optimal take-off and landing points was solved using Particle Swarm Optimization (PSO) algorithm.

Further extensions have considered routing of fuel constrained UAV and recharging on ground vehicles that are free to move on prescribed paths. This is a challenging problem because each of these vehicles have different constraints on speed and fuel capacity. Maini et al. [1] considered the problem of routing a single fuel-constrained UAV to a set of missions while being recharged by stopping at a UGV traveling on a road network. They solved the problem using a two-stage approach. First, using the UAV range constraints, they found a set of recharging depots. Second, they formulated a mixed-integer linear program and solved for the path of both the UAV and UGV. Mathew N. et al. [29] presented a cooperative rendezvous

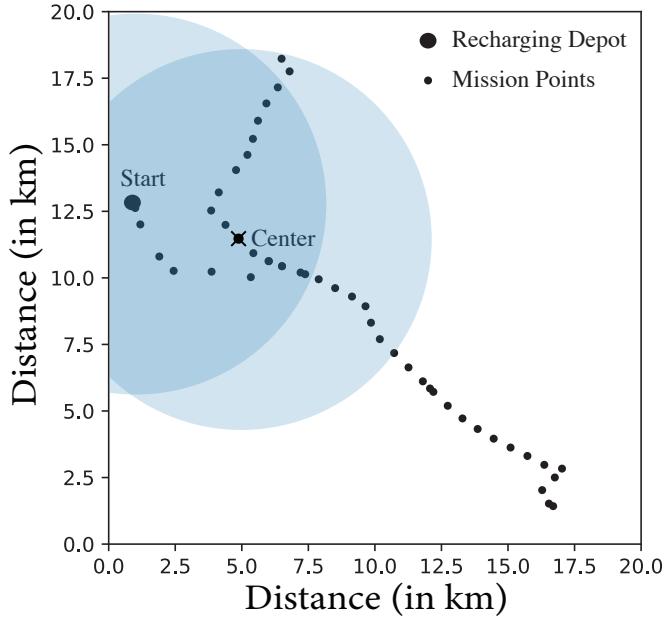


Figure 4. The problem scenario. The UAV and UGV, both start from the recharging depot. The mission points are shown with black dots. The UAV range is shown with a blue circle.

planning of working robots (UAV) and one or more mobile charging robots to recharge UAVs by formulating a rendezvous scheduling problem as a Multi Generalized Traveling Salesman Problem (MGTSP) and then transforming it into a TSP for applying heuristic solvers. The authors then extended this problem for longer planning using receding horizon strategies.

We have considered extension of the problem by considering multiple fuel-constrained UAVs and a single UGV [30]. We solved the problem in a tiered fashion. First, we use K-mean clustering to create nodes for UGVs to visit and solved for the UGV path using a traveling salesman formulation. Second, using the UGV path, we formulated and solved a vehicle routing problem with capacity constraints, time windows, and dropped visits.

One of the issues using heuristics for solving the routing problems is that the heuristics have parameters that can affect the quality of the solution. Thus, some past work has considered tuning of the parameters to improve the solution. Huang et. al. [31] solved a capacitated arc routing problem using hierarchical decomposition to generate feasible solution and then used local search. A bayesian optimization was used to improve the parameters of the hierarchical decomposition. Henrio et. al. [32] considered the problem of reducing the time between consecutive visits of a series of locations to reduce the uncertainty. They used firefly algorithm which is based on flashing of fireflies to attract or mate with other fireflies. A bayesian optimization was used to tune the parameters of the firefly algorithm. Pilat [33] used genetic algorithms to improve the parameter selection in an ant colony optimization algorithm to solve the traveling salesman problem.

In this paper, we extend our past work on UAV-UGV routing using heuristics [30]. We investigate the use of genetic algorithm and bayesian optimization to improve the heuristics. The novelty of this study lies in the application of the global optimization techniques like Genetic Algorithm and Bayesian Optimization for parameterizing different heuristics parameters which can be used for solving combinatorial optimization problems in a shorter period of time. This study shows that with proper tuning of the routing parameters it is possible to obtain a larger route coverage for tiered routing of heterogeneous vehicles. The flow of the paper is as follows. We present details about the optimization method in Sec. 2. The results are in Sec. 4, followed by the Discussion in Sec. 4. Finally, the conclusion and future work is in Sec. 5

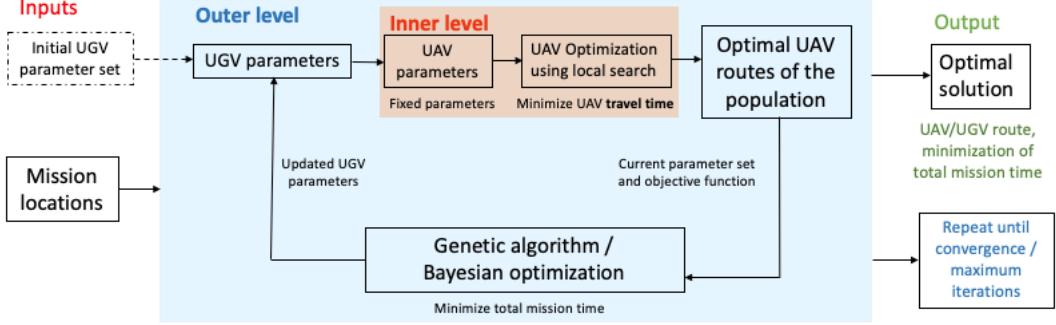


Figure 5. Overview of the algorithm

2 METHODS

2.1 Problem statement

Figure 4 shows the problem scenario considered in the paper. The mission points are shown with black dots. There is a recharging depot shown as a blue circle encompassing the solid black dot. The UAV can travel on the UGV or fly by itself. The UAV may be charged on the UGV or the depot. Both, UAV-UGV have to start and end at the depot and have to visit all the mission points.

The blue circles ($radius = fuel\ capacity/2$) of Figure 4 represent the range of the UAV on a full charge; the distance that the UAV can cover is the diameter of the circle. If the UAV starts from the center of the circle on a full charge, it can return back to the center of the circle with an empty charge. We have drawn two circles which are centered approximately at (1, 12.5) and (5, 12). It can be observed that from the start location, the UAV cannot travel to the two mission points approximately at (7.5, 17.5). However, if the UAV starts from the point

(5, 12) with a full charge, it can cover the two mission points and return back. To enable this solution, the UAV would need to ride with the UGV till (5, 12), then visit the mission points and get refueled. This illustrates some of the intricacies of choosing an appropriate path for the UGV such that the UAV can successfully move to the mission points at the extreme ends and increase the route coverage. In this problem, we tried to optimally parameterize the UGV parameters like the rendezvous points and time periods of UAV in the UGV path with the help of global optimization techniques.

2.2 Solution approach

Figure 5 shows the solution approach that involves an outer- and an inner-level. The outer level block (light blue) involves heuristics to choose a UGV route. The UGV route heuristics has a few free parameters. Once these parameters are set, the inner-level block (light orange) is the UAV route optimization. Thus far, the UAV-UGV route selection is open-loop since the UGV route has not been optimized. Thus, we run an optimization on the outer loop that optimizes the free parameters in the UGV heuristics minimizing an appropriate cost, thus closing the loop. This outer-inner loop optimization proceeds till the maximum iteration limit is reached or the solution has converged. That is, there is no change in the objective value. We now describe the details of the inner-loop, the outer-loop, and the closed-loop optimization.

2.3 Heuristics for UGV (Outer-loop)

Our heuristics for UGV route are based on maximum fuel range of the UAV described earlier (range is show as a blue circle in Figure 4). Figure 6 shows the heuristics for the UGV route. The UGV starts at the depot and travels along the mission points as shown by the arrow. Next,

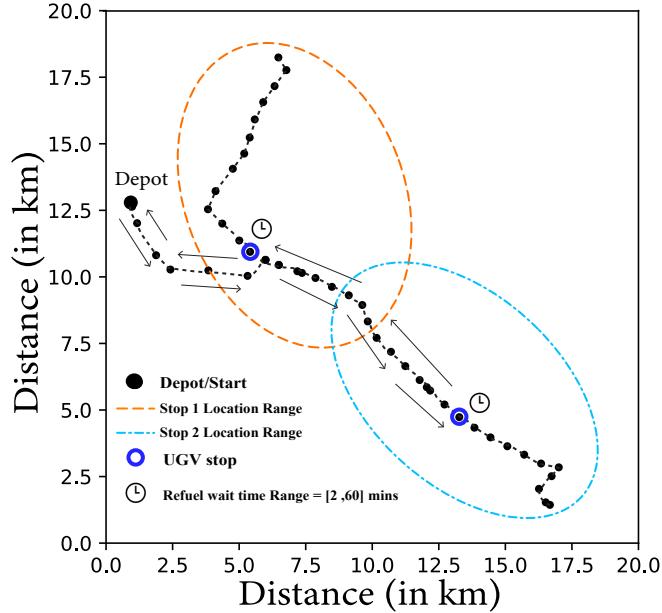


Figure 6. Heuristics for UGV route. The UGV can make two stops on any mission points such that the first one inside the red ellipse shown with dashed lines and the second is inside the blue ellipse shown with dash-dotted lines. At each stop, the UGV can choose a time to wait (e.g., to recharge the UAV).

the UGV is allowed to stop anywhere in the ellipse with dashed red lines for a prescribed time. The rationale is that in choosing a stop and wait time is to give the UAV enough time to land and recharge on the UGV. Next, the UGV moves to the bottom right side and can take another stop anywhere inside the blue ellipse with blue dash-dot lines. We have shown two random stop locations in each ellipse with a blue hollow circle. For the chosen stop locations and wait times for each stop, the UAV routing problem is formulated and solved as described in the next section.

2.4 Optimizing UAV route (Inner-loop)

We formulate a Vehicle Routing Problem (VRP) with capacity constraints to account for fuel limits, time windows to allow for rendezvous, and dropped visits to allow the UAV to visit some of the many vertices on the UGV path. We constrain the UAV to a fixed speed, pre-specify the battery capacity and service time as the UAV lands and waits on the UGV. The set of all UGV waypoints is denoted by $D = \{0, 1, 2, \dots, m\}$. There are $n - m$ pre-specified mission points which belong to the set $M = \{m + 1, \dots, n\}$. The set of all vertices is then $V = M \cup D = \{0, 1, 2, \dots, m, m + 1, \dots, n\}$. The set of all edges denotes all possible connections between the vertices $E = \{(i, j) | i, j \in V, i \neq j\}$. Consider a directed graph $G = (V, E)$ where V is the entire set of vertices $V = \{0, 1, 2, \dots, m, m + 1, \dots, n\}$ and E is the set of edges that gives the arc costs between i and j and $E = \{(i, j) | i, j \in V, i \neq j\}$. Let c_{ij} be the non-negative arc cost between a particular i and j . In this problem, the cost will be the time traveled between two nodes 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. We formulate the VRP problem with fuel constraints, time windows, and dropped visits as follows.

The objective is Eq. Equation 3.10 is to minimize the total time traveled by all the UAVs. Constraints in Eq. Equation 2.3 and Eq. Equation 2.4 represent the flow conservation where the inflow of a certain UAV should be equal to the outflow of that UAV at any vertex among the mission vertices M . Constraints in Eq. Equation 2.5 and Eq. Equation 2.6 denote the optional stops the UAV can take on the UGV vertices D , i.e., dropped visits. Next, constraint in Eq. Equation 2.7 also represents the flow conservation but here it is represented

for start and end vertices, where the number of UAVs leaving the start vertex must be equal to the number of UAVs reaching the end vertex. The start vertex and end vertex correspond to the first and last vertex of the UGV route. The constraint in Eq. Equation 3.11 is the Miller-Tucker-Zemlin (MTZ) formulation [18] for sub-tour elimination. MTZ constraint takes care of the sequential visit of each node by keeping track of values like fuel capacity, travel time of UAV corresponding to each node. It makes sure that if a node is visited twice, then the constraint is violated. This constraint enables that the UAV's energy is not fully drained out while eliminating loops. In this constraint, L_1 denotes a large number. This constraint becomes active only when there is a flow between vertices i and j and drains the UAV energy based on time taken from the two vertices. The P_{UAV} in the constraint represents the power profile of the UAV, which basically tells the power consumption when the UAV travels from one node to another. In this problem, such a power profile for the UAV is given by the following equation.

$$P_{UAV} = 0.046v_a^3 - 0.583v_a^2 - 1.876v_a + 229.6 \quad (2.1)$$

where v_a corresponds to the velocity of UAV. In this problem, the velocity of UAV is fixed to $v_a = 10$ m/s.

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \text{ such that} \quad (2.2)$$

$$\sum_{i \in V} x_{ij} = 1, \quad \forall j \in M \setminus D \quad (2.3)$$

$$\sum_{j \in V} x_{ij} = 1, \quad \forall i \in M \setminus D \quad (2.4)$$

$$\sum_{i \in V} x_{ij} \leq 1, \quad \forall j \in D \setminus \{0\} \quad (2.5)$$

$$\sum_{j \in V} x_{ij} \leq 1, \quad \forall i \in D \setminus \{0\} \quad (2.6)$$

$$\sum_{j \in V} x_{0j} = \sum_{i \in V} x_{im} = 1, \quad \{0, m\} \in D \quad (2.7)$$

$$f_j \leq f_i - (P_{UAV} c_{ij} x_{ij}) + L_1(1 - x_{ij}), \quad \forall i \in V, j \in V \setminus D \quad (2.8)$$

$$f_j = Q, \quad \forall j \in D \quad (2.9)$$

$$0 \leq f_j \leq Q, \quad \forall j \in V \quad (2.10)$$

$$t_j \geq t_i + (s_i + c_{ij} x_{ij}) - L_2(1 - x_{ij}), \quad \forall i \in V, j \in V \quad (2.11)$$

$$t_j^l \leq t_j \leq t_j^u, \quad \forall j \in V \quad (2.12)$$

$$x_{ij} = 0, \quad \forall i \in D, \forall j \in D \quad (2.13)$$

$$x_{ij} = 1 \rightarrow f_i \geq P_{UAV} c_{ij}, \quad \forall i \in V \setminus D, \forall j \in D \quad (2.14)$$

$$x_{ij} = 1 \rightarrow f_i = Q, \quad \forall i \in D, \forall j \in V \setminus D \quad (2.15)$$

$$x_{ij} = 1 \rightarrow \sum_{i \in V \setminus D} x_{ji} = 1, \quad \forall j \in D, \forall i \in V \setminus D \quad (2.16)$$

$$x_{mj} = 0, \quad \forall m \in D, \forall j \in V \quad (2.17)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in V \quad (2.18)$$

$$f_i > 0, \quad f_i \in \mathbb{R}_+, \quad \forall i \in V \quad (2.19)$$

$$t_i > 0, \quad t_i \in \mathbb{Z}, \quad \forall i \in V \quad (2.20)$$

$$s_i \geq 0, \quad s_i \in \mathbb{Z}, \quad \forall i \in V \quad (2.21)$$

$$Q > 0, \quad Q \in \mathbb{R}_+ \quad (2.22)$$

$$L_1, L_2 > 0, \quad L_1, L_2 \in \mathbb{R}_+ \quad (2.23)$$

Constraint Eq. Equation 3.12 states that if the vertex is a recharging UGV stop, then UGV has to refuel the UAV to its full capacity Q . Constraint Eq. Equation 3.13 is the condition that the UAV's fuel at any vertex in V should be between 0 and maximum fuel capacity. Constraint Eq. Equation 3.14 denotes that the cumulative arrival time at j^{th} node is equal to the sum of cumulative time at the node i , t_i , the service time at the node i , s_i , and the travel time between nodes i and j , $c_{ij}x_{ij}$. Here L_2 denotes a large number which helps to eliminate sub-tour constraints similar to Eq. Equation 3.11. The s_i in our problem is a decision variable in cases when the UAV travels from a refuel node. That is, the service time is basically the recharging time when it travels from a refuel node. In our problem, the recharging time depends on the existing charge present in the UAV. That is, if the fuel level is low it will take more time to recharge. Hence, the profile of the power transfer from UGV to UAV should be

taken into account based upon the fuel level present in UAV. A first order approximation of the battery recharge rate, P_r is given by:

$$P_r = \begin{cases} 310.8; & E \leq 270.4 \text{ kJ}, \\ 17.9(287.7 - E); & 270.4 < E \leq 287.7 \text{ kJ}. \end{cases} \quad (2.24)$$

where E represents the existing energy level on the UAV when it docks to recharge. The power transfer uses constant current until 94% of the battery capacity, with a 3.5C charge rate. After 94% capacity, it switches to a constant voltage charge. And from the existing fuel level on UAV and computing this battery recharge rate, when the UAV is charged to its maximum capacity, we would get the recharging time of the UAV. Constraint Eq. Equation 3.15 is the time window constraint that tells the vehicle to visit a certain vertex in the specified time window for that node. In this problem, the mission nodes are not constrained by time as the UAVs have the liberty to visit those mission points that benefits them according to the travel of UGV. This means that whenever UAV needs to get refueled, it would be easier for it to go to the UGV to refuel. Constraints Eq. Equation 2.13 restricts that the two consecutive visits made by the UAVs should not be consecutive UGV stops. Constraints Eq. Equation 2.14 - Eq. Equation 3.16 represents the indicator constraints where the constraints to the right side of the arrow should hold if the binary decision variable x_{ij} is equal to 1. If x_{ij} is equal to zero, then the constraints to the right side of the arrow may be violated. The constraint in Eq. Equation 2.14 tells that if there is travel from any mission vertex i to the UGV vertex j , then fuel level at the i^{th} node should be atleast equal to the energy consumed by the UAV

when it travels from i to j . Constraint in Eq. Equation 2.15 tells that if there is travel from the UGV vertex i to any mission vertex j , then the fuel level at the i^{th} node should be the maximum fuel capacity of the UAV as it is recharging to its full capacity at the UGV stop. The constraint in Eq. Equation 3.16 makes sure that if any UAV comes to the refuel vertex to recharge, then there must exist an arc between that refuel node and a mission node to maintain the flow conservation. Constraint in Eq. Equation 2.17 denotes that there should not exist any flow once the vehicle has reached the end node m . Eq. Equation 2.18 is a binary decision variable that is responsible for flow between the edges. Eq. Equation 2.19 represents the continuous decision variable that monitors the fuel level at any node and has zero as the lower bound value. Eq. Equation 2.20 represents the integer decision variable that computes the cumulative time of UAV's route and has zero as the lower bound. Eq. Equation 2.21 denotes the service time at the respective nodes, which is a positive integer with a lower bound equal to zero. Eq. Equation 2.22 represents the maximum fuel capacity of a UAV Finally, Eq. Equation 2.23 denotes the large numbers used in the constraints Eq. Equation 3.11 and Eq. Equation 3.14. The above Mixed Integer Linear Programming (MILP) formulation for UAV routing can be solved using solvers like Gurobi Optimizer [19]. For each set of UGV routing parameters, it takes a lot of time to obtain the corresponding optimal UAV routing solution using this solver. Hence, we resorted with other solving methods that gave quality inner-level solutions in a shorter period of simulation time.

Apart from UAV, the UGV has has a certain limit on its fuel capacity. The power curve for the UGV in this work follows the following equation Equation 2.25, where UGV velocity

v_g is in m/s and power is in watts. And irrespective of the nature of power profiles for UGV and UAV, this formulation helps to find the appropriate optimal solution for this co-operative vehicle routing problem. In this study only the kinematics model of the UAV and the UGV was considered.

$$P_{UGV} = 464.8v_g + 356.3 \quad (2.25)$$

2.5 Solution using Constraint Programming (CP)

We used Googles OR-ToolsTMfor implementation of the heuristics to generate the results in this paper [34] mainly for its speed of solution. Whereas Genetic algorithm and Bayesian optimization were implemented through manual Python programming for optimal tuning of the heuristics parameters in order to improve the solution quality. OR Tools uses Constrained Programming (CP) [35,36] to solve TSP and VRP problems. Constraint programming or constraint optimization is a tool for solving hard combinatorial optimization problems by searching for solutions that satisfy a list of constraints.

OR-ToolsTMuses a search tree, local search, and meta-heuristics to find feasible and, subsequently, the most optimal solutions. At the heart of OR-toolsTMis a CP-SAT solver [34]. The solver uses DecisionBuilder that has as its input, the decision variables, rules to choose the next variable to assign a value, rules for choosing the value to assign to the variable. Using the DecisionBuilder, we use the Path Cheapest Arc strategy to find an initial feasible solution (see algorithm in [37]). Starting with the “start” node, the decision builder connects the node that

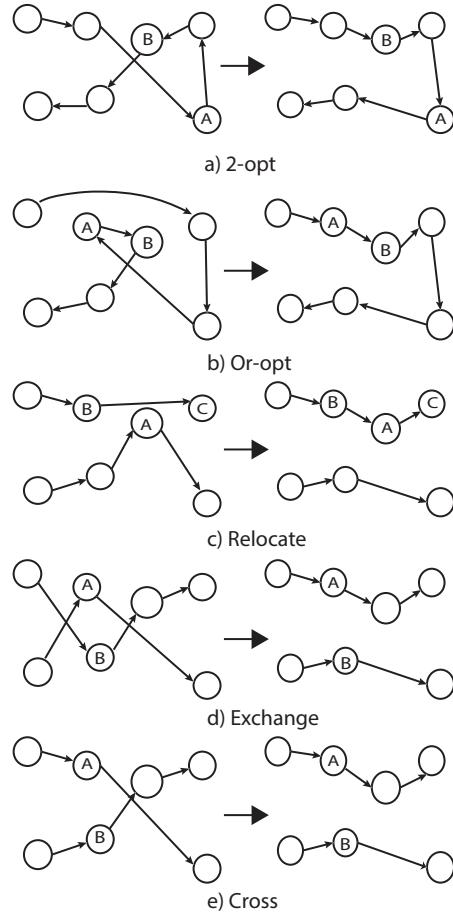


Figure 7. Move operators using in Constraint Programming [2]

has the shortest distance from the previous node and iterating till the end. While doing the connections, it checks the feasibility of the solution.ă

Then OR-ToolsTMuses a local search to find the best solution in the neighborhood of the current solution.ă

This local search proceeds by a move operator that rewires the nodes and checks for feasibility and cost. These moves are repeated until a termination criteria, such as no improvement of the objective. There are 5 move operators. These are listed next and shown in Figure 7 and is taken from [2].

1. **2-opt** interchanges the sub-part of a tour by removing two arcs, and then connects them interchangeably so that the objective value gets reduced.
2. **Or-opt** moves the sub-part of a tour if there are a maximum of 3 contiguous visits to that sub-part of the tour.
3. **Relocate** connects a visit of one tour to another tour if the reduction in objective value is seen.
4. **Exchange** involves swapping two visits between each other from either the same tour or two different tours.
5. **Cross** involves exchange of a visit at the end of one to another tour. The difference between Exchange and Cross is that the Exchange move can be done in any part of tour/tours, but Cross can be done only to the end portions of two tours.

In order to escape a local optimum solution, OR-Tools™use Guided Local Search (GLS) meta-heuristics [38].In GLS, we add a penalty term to the objective function O leading to an

augmented objective O' function. The penalty term is dependent on the neighborhood of the solution x through a set of features F . The augmented objective function is [2]

$$O'(x) = O(x) + \lambda \sum_{i \in F} f_i(x) p_i c_i \quad (2.26)$$

where the indicator function for the corresponding feature i that belongs to F is f_i . We define $f_i(x) = 1$, if the feature i is in solution or 0 otherwise. Also, λ is the penalty factor that can tune the search for the solutions. For example, a larger λ increases the diversity of the solutions (also see [39]), p_i is the number of times the particular feature i has been penalized, and c_i is the cost for the feature f_i . Using the augmented objective O' increases the cost of the objective with respect to the neighborhood, thus enabling the solver to get unstuck from a local optimum solution. Subsequently, a local search is used to continue the search.

2.6 Optimizing the parameters of the UGV heuristics

We have optimized the UAV route for a pre-selected UGV route. It is clear that changing the UGV route will change the UAV solution and consequently the optimum. In this section, we are interested in a closed-loop optimization where we would like to optimize both, the UGV and UAV solutions, thus achieving better solution. We choose 6 parameters in our UGV heuristics; the x- and y-coordinate of each of the two stop locations and the wait times at the stops. We use genetic algorithm and bayesian optimization to optimize these 6 parameters.

Algorithm 1 Genetic Algorithm

Input: Population size n , Maximum generations MAX **Output:** Global best solution

```

1 Generate initial population of  $n$  chromosomes randomly
  Set the current generation  $g \leftarrow 0$ 
2 while  $g < MAX$  do
3   if  $g = 0$  then
4     Compute the fitness value for each chromosome
      Increment the current generation  $g \leftarrow g + 1$ 
5   else
6     Select a pair of chromosomes from the initial or old population based on fitness
       Apply crossover operation on selected parents
       Apply mutation operation on produced offspring with a mutation probability
       Replace initial or old population with newly generated population
       Compute the fitness value for each chromosome
       Increment the current generation  $g \leftarrow g + 1$ 

```

2.6.1 Genetic Algorithm

Genetic algorithm is a metaheuristic inspired from the process of natural selection in nature.

It is an effective method to solve for global optimization problems where the objective function could potential have multiple local optimal solutions [40].

Algorithm 1 describes the workflow of the Genetic algorithm. In our problem, the population in each generation basically consists of the parameter set, in which each individual of the population is a list of parameters that constitute the UGV route and each parameter value in the parameter list are encoded to form genes. The genes, which are the encoded version of the parameters, are concatenated together into a single string to form an individual of the population. In technical terms, those individuals of the population are called chromosomes. We use

binary encoding for gene encoding as it helps in improving the diversity of the solutions. We use Latin Hypercube Sampling (LHS) to sample the initial population. LHS is a sampling technique that is not purely based on random sampling, but mimics some structure in randomness. That is, LHS has a memory of previously sampled points and the same sample points or same combination of points are not sampled again unlike random sampling, where repeated sampling of same points can happen. LHS mimics the distribution of the data and provides an efficient sample [41]. Once the initial population is formed, the fitness function, which is the objective function is computed for each individual. The individual in the population with better fitness are carried forward to the next generation. This is called elitism and it ensures that solutions with better fitness values will be retained. These solutions with then participate in a selection process to further improve the fitness. Thus process is repeated iteratively to improved the fitness till there are no more improvements indicating convergence. Selection process is carried on to select two individuals (parents) from the previous generation to produce offsprings for the next generation. We use a technique called *Tournamentselection* for selecting the chromosomes as it performs better than other techniques in terms of algorithm workflow, convergence rate and time complexity [42]. In this selection, two individuals to be compared are picked from the population and the individual with the best solution amongst the two will be chosen as a parent for the crossover operation. In order to extend the possibility of obtaining a global optimal solution, a variant in tournament selection called *unbiasedtournamentselection* is introduced as it eliminates the loss of diversity related to the failure of random sampling for tournament selection [43]. Instead of picking two individuals at random for comparison, the individuals

are picked in a particular order viz., permutation and then compared. The two next steps are the crossover and mutation operations, help to improve the solution search space which helps achieve a global optimum. In crossover operation, the selected pair of parents are mated at a randomly chosen crossover points to produce offsprings. For this problem, 2-point crossover operator is used to produce two offsprings from the parents. After performing the crossover operation, the mutation operation is performed in which a random bit of the offspring's chromosome is flipped or mutated if the probability of the random bit exceeds a certain probability value to impart diversity in the solution. This diversity enables the solution to escape the local optima. The probability of the random bit is taken from the uniform distribution. For this problem, the mutation operator value of the GA is set to be 0.01. The GA algorithm is terminated if there is no improvement in the solution, indicating convergence or when the maximum iteration limit is reached.

2.6.2 Bayesian Optimization

Bayesian Optimization is a powerful tool for optimizing computationally expensive objective function. Mathematically, bayesian optimization acts as a global optimizer of a blackbox function $f(x)$: $x^* = \arg \min_{x \in \mathcal{X}} f(x)$, where \mathcal{X} is the region of interest for finding the optimal solution. Bayesian optimization uses a probabilistic model (surrogate model) $f(\cdot)$ based on given prior distribution $\mathcal{D}_n = [(x_1, f(x_1)), (x_2, f(x_2)), \dots, (x_n, f(x_n))]$ for analyzing its posterior belief at the unexplored input regions. The performance of Bayesian Optimization depends significantly on the acquisition function which balances between the exploitation and exploration of the predictive surrogate model $f(\cdot)$ to list out the most promising points x^+ in its domain

\mathcal{X} and evaluates the objective at these points, $f(x^+)$. Next $(x^+, f(x^+))$ is added to the prior distribution of the surrogate model $f(\cdot)$ and consequently the posterior distribution is updated. This process is continued iteratively till the solution converges to its optimum value or when maximum iteration limit is reached.

Algorithm 2 shows the pseudo code for Bayesian Optimization. Bayesian optimization has two major blocks, first is the surrogate model $g(x)$ which approximates the objective function $f(x)$ based on the sampling points \mathcal{D}_n and the second, the acquisition function α which helps to evaluate the important points (x_{n+1}, y_{n+1}) in the posterior. We have used Gaussian prior as the surrogate model and Expected Improvement (EI) for the acquisition function. Suppose, $\mathcal{GP}(f(\hat{x}), s^2(x))$ is the surrogate model on $f(x)$ and f_{min} is the minimum value of the objective functions among $[f(x_1), f(x_2), \dots, f(x_n)]$ for n given values of $[x_1, x_2, \dots, x_n]$. The improvement $I(x)$ of $f(\tilde{x})$ on unexplored points \tilde{x} is defined as

$$I(x) = \max(f_{min} - f(\tilde{x}), 0) \quad (2.27)$$

Thus, Expected Improvement (EI) acquisition function is the expectation (average) of the variable $I(x)$ over $f(x)$ defined as below:

$$\begin{aligned} E(I(x)|f(x)) &= E[\max(f_{min} - f(\tilde{x}), 0)|f(x)] \\ &= (f_{min} - f(\hat{x}))\Phi\left(\frac{f_{min} - f(\hat{x})}{s(x)}\right) \\ &\quad + s(x)\phi\left(\frac{f_{min} - f(\hat{x})}{s(x)}\right) \end{aligned} \quad (2.28)$$

Algorithm 2 Bayesian Optimization

Input: Sampling points \mathcal{D}_n , Maximum iterations k **Output:** Global best solution

- 7 Make $g(x)$ surrogate model on \mathcal{D}_n
 - 8 **for** $k = 1, 2, \dots$ **do**
 - 9 Obtain x_{n+1} by optimizing acquisition function α

$$x_{n+1} = \arg \max \alpha(g(x)), \quad y_{n+1} = f(x_{n+1})$$

$$\text{Update } \mathcal{D}_{n+1} = [\mathcal{D}_n, (x_{n+1}, y_{n+1})]$$

$$\text{Update } g(x) \text{ on } \mathcal{D}_{n+1}$$
 - 10 Find $x^* = \arg \min g(x), \quad y^* = f(x^*)$
-

The point with maximum EI is chosen as the point of interest to update the prior \mathcal{D}_{n+1} and the surrogate model $g(x)$. The process is continued till the maximum iteration k is reached when we get the optimal point (x^*, y^*) from the Bayesian Optimization.

3 RESULTS

We present results for the scenario shown in Figure 4. The UAV-UGV have to start and end at the Depot. The mission points are shown by black dots. All missions point needs to be covered by either the UAV or the UGV. The UAV can recharge at the Depot or at the UGV. The UAV and UGV velocities when moving are fixed at 10 m/s and 4 m/s respectively. The UAV and UGV fuel capacity are 287.7 kJ and 25.01 MJ respectively.

The heuristics for the UGV are depicted in Figure 6. There are 6 free parameters for the UGV optimization. Four for the x- and y-stop locations (stop 1 and stop 2) of the UGV and 2 wait times. Table V shows the ranges for both these parameters. The objective function is to minimize the total time to visit all mission points.

Parameter	Range
UGV stop 1 location (km,km)	(6.90,18.04) to (9.80,9.07)
UGV stop 2 location (km,km)	(8.64, 9.77) to (16.96,1.45)
UGV stop 1,2 wait times (min)	2 to 60

TABLE II

Parameter set and their range for GA/BO optimization

We used Python 3 for the optimizations (GA/BO/OR-Tools) performed the computations on a 3.7 GHz Intel Core i9 processor with 32 GB RAM on a 64-bit operating system.

Table III gives the optimal parameter set computed by GA/BO. The UGV 1 stop locations are slightly different but the wait times are substantially different. The UGV stop 2 locations and times are identical; the stop 2 corresponds to the far right corner of the missions. These results are discussed in detail later in this section.

Table IV compares the key metrics between GA and BO optimizations. The objective of the optimization was to reduce the total time. Both optimization gave similar results with BO being marginally better than GA by 3 min. GA needed 3 times more local-search optimizations and took 6 times more computational time than BO. Thus, BO has more computational efficiency than GA. The UGV results show that GA and BO had similar times, energy consumed, but the UGV travel more mission points in GA rather than BO. The UAV results show substantial

Parameter	GA Values	BO Values
UGV stop 1 (km,km)	(4.99,11.65)	(6.10,10.80)
UGV stop 1 wait time (min)	18	50
UGV stop 2 (km,km)	(16.96,1.45)	(16.96,1.45)
UGV stop 2 wait time (min)	3	2.45

TABLE III
Optimal parameters after GA/BO optimization

difference between the two methods. The travel time, energy consumed in BO are about 1.5 times higher. This is because the BO travels to 3 more mission points than GA.

Figure 8 and Figure 9 shows the optimum route produced by GA and BO respectively. The main difference between the two solutions is in the UAV route. In GA, the UAV visits the missions further away from the intersection of the branches as shown in Figure 8 a), b), and c) then recharges on the UGV, completes the remaining missions as shown in d) and goes to the start. In BO, the UAV visits the missions closer to the intersection of the branches as shown in Figure 9 a), b), then recharges once on the UGV. Then the UAV visits the top most mission points as shown in c) and d) then recharges a second time on the UGV. Finally, the UAV completes the remaining missions as shown in e) and f). Thus, it can be observed that the order of choosing the mission points in BO increases the travel time of UAV. However, since

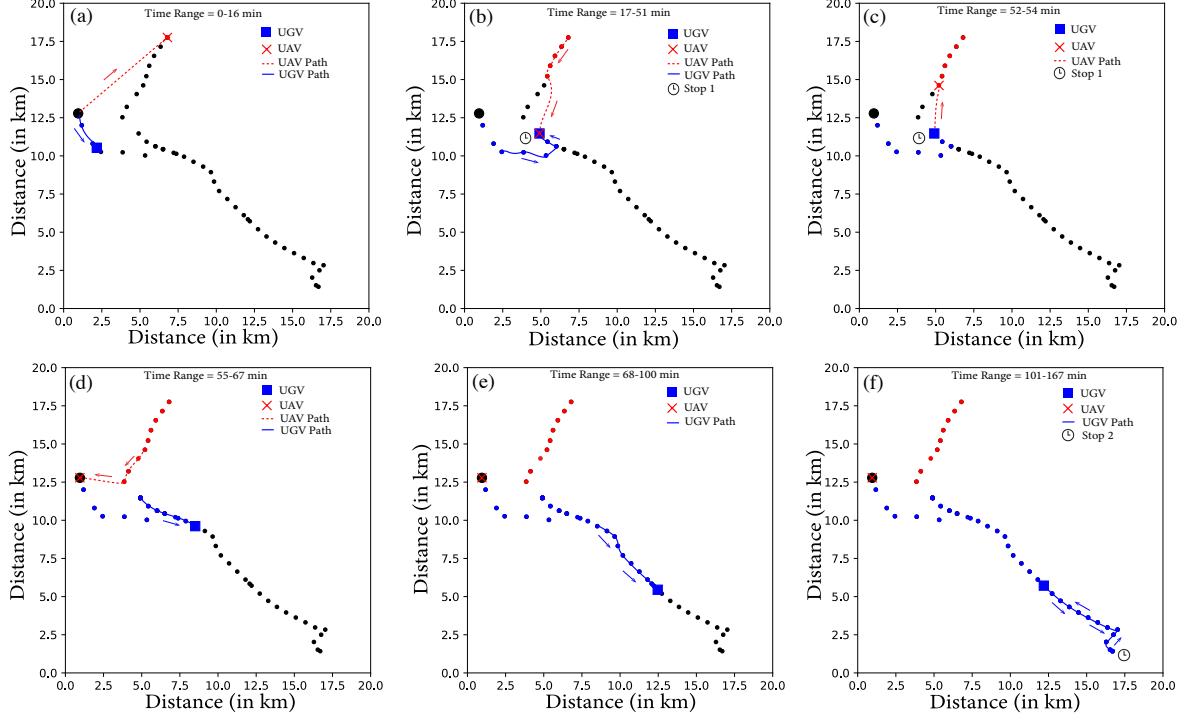


Figure 8. Solution produced by Genetic Algorithm. The plots show the UAV and UGV route at different time spans.

the UGV is the slower of the two vehicles, the travel time of the UGV determines the total time taken to cover all mission points. Since the UGV travel is almost the same except for the slightly more wait time for the UAV to return back in the BO solution, the difference in the objective is only 3 minutes.

4 DISCUSSION

In this paper, we presented a framework for optimizing routes of heterogenous vehicles, a UGV and UAV, with fuel constraints. The key idea is to solve the problem in a tiered fashion.

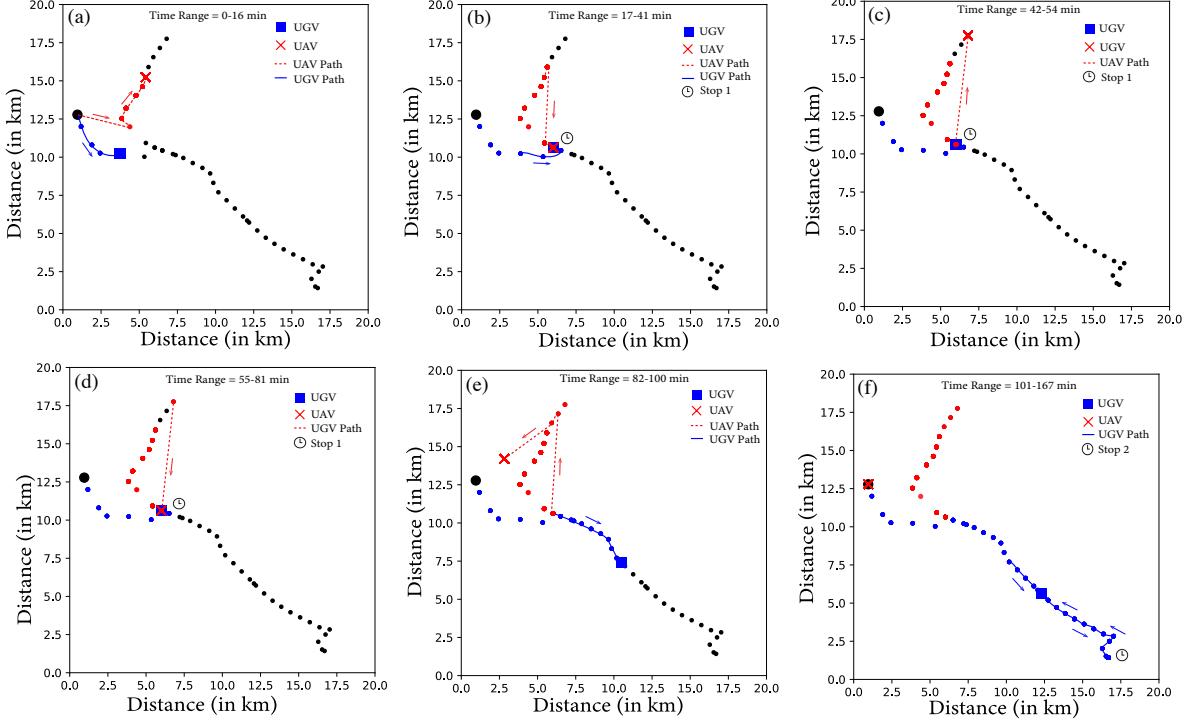


Figure 9. Solution produced by Bayesian Optimization. The plots show the UAV and UGV route at different time spans.

First, we use heuristics to decide the UGV route. Second, we optimize the UAV route using a local search. We identify key parameters in the UGV heuristics and optimize them iteratively with UAV route using genetic algorithm and bayesian optimization. Starting from an infeasible solution, both algorithms are able to give solutions with similar cost, but different solutions.

Our objective was the time taken to visit all the mission points. Both GA/BO gave almost the same time. However, the individual UAV solutions were quite different. The UAV for the BO used more energy and travelled for more time. However, since the total time was dependent

on the UGV travel time (as it was the slower vehicle) and the UGV travel route were similar, the overall cost was the same. If the UAV time, energy, and other metrics are important then they can be added to the cost function using a weighted sum.

Bayesian optimization was more efficient than Genetic algorithm in the number of function evaluations and computational time. BO is designed for computationally expensive function evaluations hence it is ideal in such cases where it takes substantial time to compute a solution by doing a local search for the UAV route.

One of the main limitations of the work is that UGV heuristics are limited to only 6 parameters which restricts the solution space. Adding more parameters will potentially make the search space too large and computationally restrictive. Another limitation is that the parameters and their range for the UGV heuristic optimization had to be chosen by trial and error.

5 CONCLUSIONS AND FUTURE WORK

We conclude that a tiered optimization is a feasible method to solve heterogenous vehicles optimization where the solution of one vehicle needs to be performed before the other one. In our case, the UGV route needs to be determined first as the UAV route involves refueling on the UGV. Moreover, by suitably parameterizing the heuristics and optimizing the parameters with global optimization methods enables exploration of the space and provide good quality solutions.

Our future work would explore both these optimization methods with different routes, different costs, and different heuristics in order to make broader claims about validity of the

proposed approach. And moreover, since this research work addresses a problem-solving based approach where we focus on solving the problem that is given in hand, the algorithm that was developed focuses on solving problems with scenario maps that are similar to the scenario used in this work rather than focusing on comparing this algorithm to benchmark instances in the literature. Future works will deal with testing this algorithm on various scenarios in order to ensure the robustness of the algorithm.

Metrics	Genetic Algorithm (GA)	Bayesian Optimization (BO)
Total time (min)	225	222
Total local-search optimizations	180	60
Computational time (min)	90	15
UGV results		
Travel time (minutes)	225	222
Energy consumed (MJ)	23.13	24.86
Mission visited	37	34
UAV results		
Travel time (minutes)	65	103
Energy consumed (kJ)	575.25	804.905
Recharging stops on UGV	1	2
Recharging stops on Depot	0	0
Missions visited	9	12

TABLE IV

Comparison between GA/BO on metrics from the optimal solution.

CHAPTER 4

SOLVING VEHICLE ROUTING PROBLEM FOR UNMANNED HETEROGENEOUS VEHICLE SYSTEMS USING ASYNCHRONOUS MULTI-AGENT ARCHITECTURE (A-TEAMS)

Overview: Fast moving but power hungry unmanned aerial vehicles (UAVs) can recharge on slow-moving unmanned ground vehicles (UGVs) to cooperatively perform tasks over wide areas. Such a cooperation can be achieved efficiently by solving a path planning problem. On top of solving a path planning problem, the problem of routing an heterogeneous set of vehicles in an optimal fashion is quite challenging. In order to solve the computationally expensive path-planning problem in a reasonable time, we created a two-level optimization approach with heuristics. At the outer level, the UGV route is parameterized by considering which set of locations to visit in the scenario and the UGV wait times to recharge UAVs and at the inner level, the UAV route is solved by formulating and solving a vehicle routing problem with capacity constraints, time windows, and dropped visits. The UGV free parameters need to be optimized judiciously in order to create high quality solutions. We explore two methods for tuning the free UGV parameters: (1) a Genetic Algorithm (GA), and (2) Asynchronous Multi-agent architecture (A-teams). The A-teams uses multiple agents to create, improve, and destroy solutions.

parts of this chapter is taken from the following published journal article:
Ramasamy, S., Mondal, M. S., Reddinger, J. P. F., Dotterweich, J. M., Humann, J. D., Childers, M. A., & Bhounsule, P. A. (2023, June). 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) (pp. 95-102). IEEE.

The parallel asynchronous architecture enables A-teams to quickly optimize the parameters. Our results on test cases show that the A-teams produces similar solutions as GA but is 2-3 times faster.

Keywords: Vehicle Routing Problem, Multi-Agent Optimization, Unmanned Systems, Combinatorial Optimization

1 INTRODUCTION

There has been a considerable increase in the use of the small Unmanned Aerial Vehicles (UAVs) across diverse fields such as entertainment and logistics [20]. The reason for such a widespread adaption is because they are agile robots that can navigate at high speeds in complex environments otherwise inaccessible to humans [44]. Although UAVs are fast, they are limited by their battery capacity to a relatively small area [22].

To complete tasks over wider areas, UAVs could be provided mobile recharging stations that are hosted by unmanned ground vehicles (UGV). Such cooperative routing of a team of UAV-UGVs have been utilized in tasks such as inspection in cluttered environments [45], congested urban environments [46] and post-disaster relief [47], [48].

The cooperative routing of UAV and mobile recharging stations is complex and computationally challenging [26]. The formulation of the problem involves minimizing a cost such as the time or fuel consumption while constraining the fuel capacity and speed limits of the UAV and UGV and ensuring that they are able to rendezvous efficiently. Although it is relatively easy to formulate the problem, it is difficult to solve the formulation using exact methods due to the combinatorial nature of the problem. However, using suitable heuristics, it is possible to achieve high quality solutions relatively quickly.

There has been a considerable work done in the literature related to solving fuel-constrained routing of UAVs. Sundar et. al., [6] worked on Fuel-Constrained UAV Routing Problem where a generalization of the asymmetric Traveling Salesman Problem (TSP) is solved using Approximation algorithm and fast heuristics. A Mixed Integer Programming Problem formulation is also proposed to obtain optimal solutions. Here, a single UAV is used and gets recharged on fixed depots. Venkatachalam et. al., [49] modeled a multiple fuel-constrained UAV routing problem with fixed recharging depots. Here, the authors implemented a two-stage stochastic optimization problem with uncertainties in the fuel consumption of UAVs. Heuristics are used to achieve high quality solutions with faster computing time.

Some extensions in the aspect of heterogeneous vehicle routing were also considered in the literature to overcome some of the limitations existed in such fuel-constrained UAV routing problem with fixed depots. Subramanian et. al. [30] considered the vehicle routing problem of multiple fuel-constrained UAVs and a single UGV that acts as a mobile recharging vehicle. The problem is being solved in a tiered fashion. The authors used K-means clustering and TSP to solve UGV routing problem, and then implemented Vehicle Routing Problem (VRP) with fuel, time and optional node constraints. The aforementioned work is extended in [17] where a more generalized approach is taken to solve several different scenarios and proved the robustness of the algorithm. The above works allows the UGV to move freely on any paths, but there are some works in the literature that considers heterogeneous UAV-UGV vehicle routing with UGV constrained to move on certain prescribed paths. This is a challenging problem because each of these vehicles have different constraints on speed and fuel capacity. Maini et

al. [1] considered the problem of routing a single fuel-constrained UAV to a set of task locations while being recharged by stopping at a UGV traveling on a road network. They solved the problem using a two-stage approach. First, using the UAV range constraints, they found a set of recharging depots. Second, they formulated a mixed-integer linear program and solved for the path of both the UAV and UGV. Safwan et. al., [50] performs a bi-level optimization on a road network with prescribed UGV paths and the obtained simulation results are validated by performing a lab-setup experiment, which asserts the ability to map from simulation setup to real-time practical deployment. The work in [51] also allows the UGV to move freely only on prescribed paths and is also followed in this paper.

Since UGV has a fixed route, the heuristics for the UGV could be modeled as a parameter tuning problem as that would provide a better solution by tuning the heuristic parameters. [31] applied Bayesian Optimization to tune the hierarchical decomposition algorithm parameters and thus helped to achieve optimal solutions at a faster rate. Although such algorithms help us to achieve globally optimal solutions, they come at a cost of significant computational time. This was seen from the previous work by the authors [51] where GA and Bayesian Optimization (BO) are implemented for parameter tuning to obtain the UGV route. The results from that work show that particularly in case of GA, higher computational time of about 180 minutes was needed to solve that problem. The reason for high computational time is that these algorithms GA and BO are basically global optimization algorithms. Although such global optimization algorithms perform search over a larger space, this compromises their efficiency. Hence they can be used when the computational time is not critical, such as offline optimization [52]. At

the same time, local optimization algorithms provide quicker solutions, but are optimal in a small region of space.

To achieve faster global optimal solutions, Sachdev [53] worked on proposing an architecture called A-Teams, which was originally developed by Talukdar et. al. [54]. In A-teams, global optimization methods search over a larger space to find potentially feasible solutions. These are then improved by the local optimization methods. The author implemented two algorithms, Stochastic Quadratic Programming (SQP), a local optimization method, and GA, a global optimization method, in A-Teams to show how the advantages of local and global optimization algorithms can be tapped to produce a better result in a computationally efficient manner. Jedrzejowicz et. al., [55] proposed A-Teams to solve a Resource Constrained Project Scheduling Problem. The authors perform Reinforcement Learning (RL) along with using other optimization algorithms like local search, tabu search to solve the problem using A-Teams. The RL component in their architecture helps to apply dynamic strategy for interaction between those different optimization algorithms in an A-Team. Those authors use a middleware called JABAT (Java Agent DEveleopment-Based A-Teams), to implement the A-Teams architecture. Kazemi et. al., [56] implemented A-Teams for solving a Production-Distribution Planning Problem where each agent in their architecture uses a GA sub-module to handle its tasks and conclude that the combined multi-agent GA system provides better solutions than the individual ones for their problem. Recent works by Jedrzejowicz et. al., [57] involves implementing this architecture to solve a Resource Investment Problem in which different agents use Local search,

Lagrangian relaxation, Path relinking algorithms, Crossover operators and cooperate together to solve such a problem.

The usage of A-Teams is also found amongst the routing problems in the literature. Rabak et. al. [58] presented the A-Teams framework to optimize the automatic electronic component insertion process on an inserting machine. They implemented a combination of Quadratic Assignment Problem and Traveling Salesman Problem (TSP) in the framework to perform optimization. The above work shows the framework's ability to handle multiple algorithms simultaneously. Such a realization becomes helpful in this work where the A-Teams come in hand for utilizing the local and global optimization algorithms, whose advantages and disadvantages were talked about a few lines before. Barbucha et. al., [59] worked on investigating the effects and impact of a Team of A-Teams working in parallel to solve difficult combinatorial optimization problems. In their work, different algorithms cooperate together within an A-Team, and several similar A-Teams are made to work in parallel. The computational efficiency of their architecture is demonstrated by solving benchmark instances in different problems like Euclidean Planar Traveling Salesman Problem (TSP), Vehicle Routing Problem (VRP), Clustering Problem (CP), and Resource Constrained Project Scheduling Problem (RCPSP). Rachlin et. al., [60] implemented the A-Teams to solve a Traveling Salesman Problem (TSP) by using Farthest Insertion and Arbitrary Insertion heuristic algorithms in their architecture.

The A-teams has been limited to solving basic routing problems (e.g., TSP). Thus, the main contribution of this work is that we use the A-teams architecture to solve a heterogeneous and co-operative vehicle routing problem involving a UAV and a UGV. We also compare the

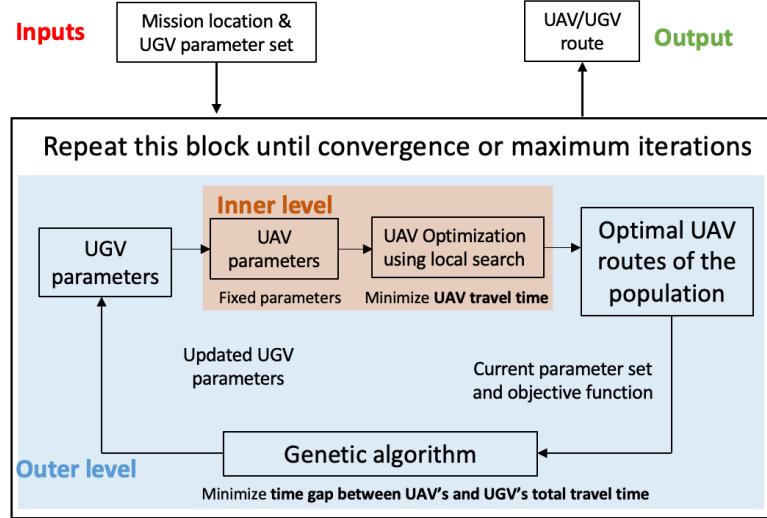


Figure 10. Overview of the bi-level optimization algorithm. The outer-level block is run in parallel on multiple cores.

A-teams architecture with results obtained using genetic algorithms in several scenarios. The flow of the paper is as follows. We present details about the optimization method in Sec. 2. The results are in Sec. 4, followed by the Discussion in Sec. 4. Finally, the conclusion and future work is in Sec. 5

2 METHODS

The authors developed a two-level optimization framework [51] that uses Genetic Algorithm and Bayesian optimization and is described in Sec. 2.1. The main contribution of this paper is the A-teams architecture which uses the two-level optimization framework as its basis and is described in Sec. 2.2.

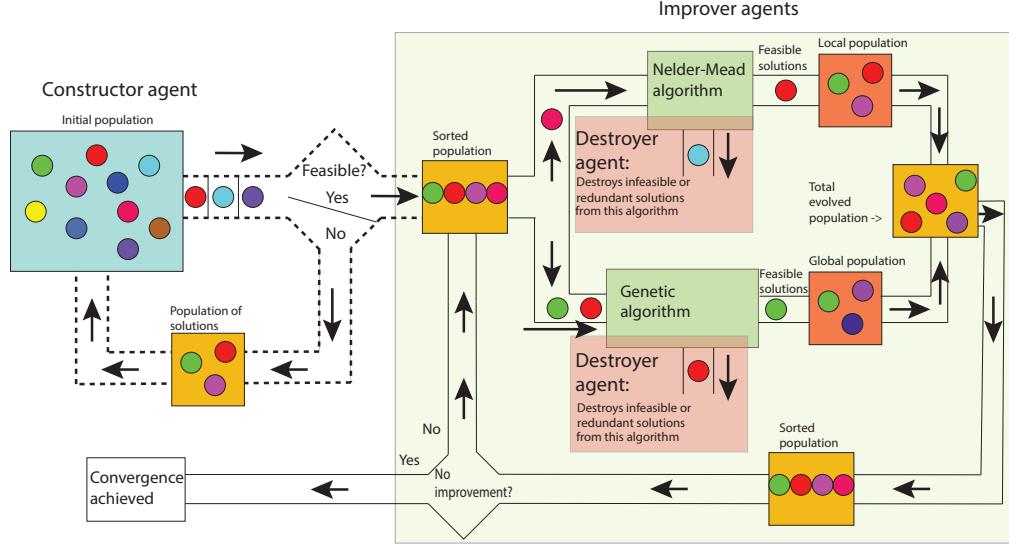


Figure 11. Implementation of A-Teams architecture for this cooperative routing problem.

2.1 Conventional Two-level optimization

The Figure 10 shows the conventional two-level optimization [51]. The outer level block shown in blue are the heuristics to choose a UGV route. The UGV route heuristics has a few free parameters. Once these parameters are set, the inner-level block shown in orange performs the UAV route optimization using Googles OR-Tools™ [34]. The UAV route optimization is heavily dependent on the free parameters of the UGV route. These parameters are optimized using a genetic algorithm. The complete block (inner/outer loop) runs several times till the genetic algorithm can no longer improve the solution or when the maximum iteration limit is reached.

2.2 Description of the proposed architecture - A-Teams

A-Teams is an architecture that uses a team of autonomous agents to perform optimization on a given problem. The agents have a common set of potential solutions. Each agent works asynchronously on the potential solutions to find better solutions which are then updated as the new potential solutions. There are three main agents that constitute this architecture and are described next.

1. Constructor Agent is used to develop an initial pool of solutions using the user inputs.
2. Improver Agent is used to improve on the pool of solutions using different optimization methods. It is important to choose complementary optimization methods (e.g., global and local optimizers) to help improve the quality of the solutions.
3. Destroyer Agent is used to discard non-optimal and bad solutions. It does this by ranking the solutions based on the cost and constraints satisfaction.

Populations are shared repositories for storing solutions computed and evaluated by different agents. These are accessible by all agents.

The architecture is modular and distributed which enables each optimizer to work independently. However, the architecture also has mechanisms to combine solutions generated by individual optimization to realize further improvements of the solutions. This makes the framework very powerful producing optimal solutions in a computationally efficient manner.

Figure 11 shows the implementation of A-Teams to solve this problem. Note that the A-teams operates over the two-level optimization block shown in Figure 10. The Constructor agent utilizes a ‘randomized’ initial UGV parameter set to construct an initial population

of solutions. The algorithm is used until the feasibility of a solution in the population is achieved. Every time the constructor agent pulls solution from that initial population, it fills up a current population, that was initially empty (represented in orange box in Figure 11), until feasibility. These solutions are usually sub-optimal, but are then passed to the Improver agent. The Improver agents improve the solutions by using two algorithms: (1) the Nelder-Mead, a gradient free direct search method, that is good for local optimization and (2) the Genetic Algorithm that is inspired from natural selection and is good global optimization method. These two algorithms are complementary in nature; Genetic algorithm searches big regions of the parameters set (exploration) while Nelder-Mead improves on the solution in the vicinity of the current solution (exploitation).

We now describe Algorithm 4 used in A-teams. On lines 1 and 2 the constructor agent role is to generate feasible solutions for the improver agent. A random initial parameter set for UGV is generated and checked if it leads to a feasible UAV solution. After a sufficient number of good feasible solutions are generated, the algorithm proceeds to the main while loop that uses the constructor and destroyer agent. When the constructor agent produces a feasible solution, the current population which has been updated so far from the initial population is sorted, and the role is handed over to the Improver agents. The solutions are sorted based on the cost and the improver agent uses the global optimizer (GA) shown on line 5 or local optimizer (Nelder-mead) shown on line 6. These improver agents work in parallel. Thus, both, the exploration and the exploitation happens simultaneously and independently. Next, on line 7, the destroyer agent looks at all the solutions and discards the infeasible solutions and those

that are already existing in the pool of solutions. Finally, on line 8, all the good solutions are pooled together and sorted in order to get ready for the next iteration. This process repeats until convergence is achieved, i.e., when there is no improvement in the population.

2.2.1 Nelder-Mead algorithm

Nelder-Mead method is a simplex-based numerical method that is used for solving the unconstrained optimization problem. Being a numerical method, it is a gradient-free direct search method and it is mainly used to solve problems in which the derivatives of an objective function are very hard to find. Since this is a simplex-based algorithm, the shape of the simplex is fixed to have $n + 1$ vertices where n is the number of input dimensions. The dimension of the simplex will increase with increase in the number of input dimensions. For example, for 2-dimensional input, the simplex will be a triangle; for 3-dimensional input, the simplex will be tetrahedron; for 4-dimensional input, the simplex will be pentachoron and so on. Once the simplex vertices have been sorted as per the objective function evaluation at each initialized random vertex, vertex x_1 will be the best solution and vertex x_{n+1} be the worst solution. Based upon this sorting, certain operations are performed for solution improvement. This algorithm involves four operators: *reflection*, *expansion*, *contraction*, *shrink*. For each operator, there is a scalar parameter which allows to perform the improvement of the solutions and the scalar parameter for those four operators are within a certain bounds that are user-defined. The description of each of the operators is as follows:

- **Reflection:** This operation is performed to make the worst vertex (x_{n+1}) of the simplex shape at least better than the second worst vertex (x_n). The reflection is performed to

mirror x_{n+1} about the centroid of the simplex shape, and the reflected point is named x_r . The x_r is found via the equation in line 8 of the algorithm Algorithm 3. Based upon the objective function evaluation at x_r , either of the following operations *expansion*, *contraction* or *shrink* is performed accordingly.

- **Expansion:** This operation is performed if the reflected solution x_r is not only better than x_n , but also better than the current best solution x_1 . If this is the case, then the solution x_r is updated into x_e by moving along that reflected vector direction, and now the new worst solution point will be x_n . The x_e is found via the equation in line 10 of the algorithm Algorithm 3.
- **Contraction:** This operation is performed if the reflected solution x_r did not get better than x_n solution. There are two types of contraction. **Outside contraction** x_{oc} is performed at a third-quarter of the distance between x_{n+1} and x_r and **Inside contraction** x_{ic} is performed at a quarter of the distance between x_{n+1} and x_r . Comparing these two contractions, x_{n+1} will get updated to x_{ic} or x_{oc} by picking the better solution between them. The x_{oc} or x_{ic} is found via the equation in line 12 or line 14 of the algorithm Algorithm 3.
- **Shrink:** If none of the above operations improve the solution x_{n+1} , then this operation is performed where it updates the current vertices x_2 till x_{n+1} to a location as per the equation in line 15 of the Algorithm 3.

One thing to keep in mind is that, the first three out of four operations above are performed on the worst input variable x_{n+1} by updating this variable values to the ones after performing

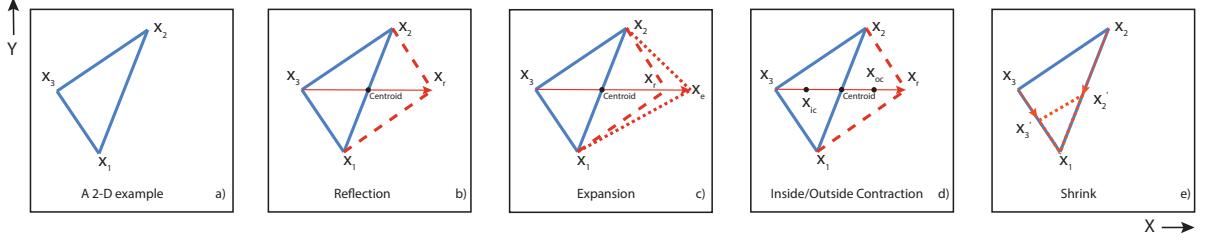


Figure 12. Graphical representation of the Nelder-Mead algorithm for 2-dimensional input example. a) The example inputs under consideration. b) Reflection operation. This operation is performed to make the worst vertex (x_3) of the simplex shape at least better than the second worst vertex (x_2). The reflection is performed to mirror x_3 about the centroid of the simplex shape, and the reflected point is named x_r . The x_r is found via the equation in line 8 of the algorithm Algorithm 3. Based upon the objective function evaluation at x_r , either of the following operations *expansion*, *contraction* or *shrink* is performed accordingly. c) Expansion operation. This operation is performed if the reflected solution x_r is not only better than x_2 , but also better than the current best solution x_1 . If this is the case, then the solution x_r is updated into x_e by moving along that reflected vector direction, and now the new worst solution point will be x_2 . The x_e is found via the equation in line 10 of the algorithm Algorithm 3. d) Contraction operation. This operation is performed if the reflected solution x_r did not get better than x_2 solution. There are two types of contraction. **Outside contraction** x_{oc} is performed at a third-quarter of the distance between x_3 and x_r and **Inside contraction** x_{ic} is performed at a quarter of the distance between x_3 and x_r . Comparing these two contractions, x_3 will get updated to x_{ic} or x_{oc} by picking the better solution between them. The x_{oc} or x_{ic} is found via the equation in line 12 or line 14 of the algorithm Algorithm 3. e) Shrink operation. If none of the above operations improve the solution x_3 , then this operation is performed where it updates the current vertices x_2 and x_3 to a location as per the equation in line 15 of the algorithm Algorithm 3.

them and the second worst variable x_n will be updated as the current worst variable after that operation. Figure 12 describes the Nelder-Mead algorithm in graphical form for a 2-input dimension case where $f(x_1) < f(x_2) < f(x_3)$. Since it's a 2D case, the simplex used is triangle. where a) represents the reflection operation where variable x_3 is updated to x_r and thus x_2 becomes x_3 , b) represents the expansion where x_3 is updated to x_e based upon

the corresponding condition in the algorithm, c) represents the outside and inside contraction operation and in actual optimization, either one of those two contractions are applied based upon the function evaluation of the reflected solution, and d) represents the shrink operation.

2.3 Heuristics for UGV (Outer-level)

Our heuristics for UGV route are based on maximum fuel range of the UAV described earlier (range is shown as a blue circle in Figure 13). Figure 14 shows the heuristics for the UGV route. The UGV starts at the depot and travels along the task locations. Next, the UGV is allowed to stop anywhere in the ellipse with dashed red lines for a prescribed time. The rationale is that in choosing a stop and wait time is to give the UAV enough time to land and recharge on the UGV. Next, the UGV moves to the bottom right side and can take another stop anywhere inside the blue ellipse with blue dash-dot lines. We have shown two random stop locations in each ellipse with a blue hollow circle. There are 7 parameters for the UGV heuristics; the starting location of the UGV/UAV, the x- and y-coordinate of each of the two stop locations, and the wait times at the stops.

2.4 Optimizing UAV route (Inner-level)

We formulate a Vehicle Routing Problem (VRP) with capacity constraints to account for fuel limits, time windows to allow for rendezvous, and dropped visits to allow the UAV to visit some of the many vertices on the UGV path. We constrain the UAV to a fixed speed, pre-specify the battery capacity and service time as the UAV lands and waits on the UGV. Constrained Programming approach is being used to solve this VRP using OR-Tools solver.

The mathematical details are not included here because of space constraint, but can be found in [51].

3 RESULTS

We used Python 3 for all the computations: a custom-written genetic algorithm and Nelder Mead from Scipy package for UGV parameter optimization, and OR-tools for UAV optimization. All computations were done on a 3.7 GHz Intel Core i9 processor with 32 GB RAM on a 64-bit operating system.

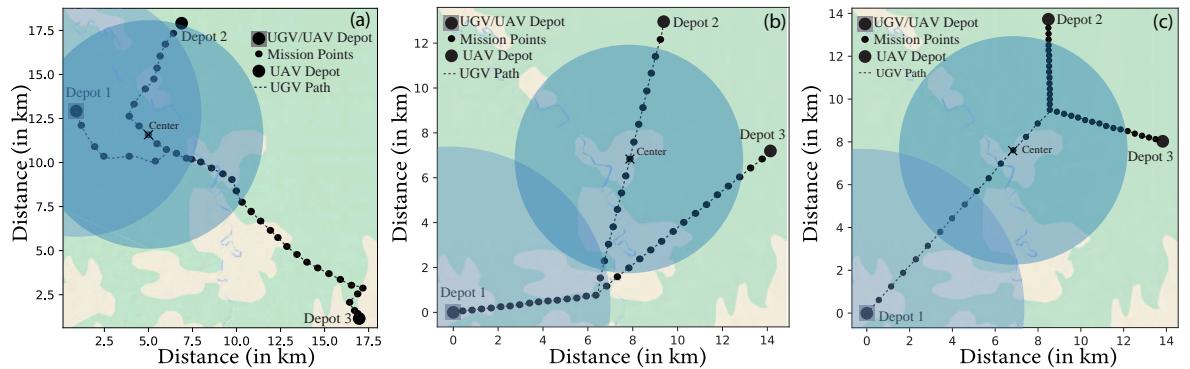


Figure 13. Description of different scenarios. The UAV and UGV, both start from one of the recharging depots. The task locations are shown with black dots. The UAV range is shown with a blue circle. a) Scenario 1 b) Scenario 2 c) Scenario 3.

Figure 13 shows the problem scenarios considered in this paper. The task locations are shown with black dots. There are 3 recharging depots shown with a black dot that is bigger

than the one used for task locations. The UAV can travel on the UGV or fly by itself. The UAV may be charged by the UGV or at the depot. Both, UAV-UGV start and end at the depot.

The blue circles represent the range of the UAV on a full charge; the distance that the UAV can cover is the diameter of the circle. For example, consider Figure 13 (a). If the UAV starts from the center of the circle on a full charge, it can return back to the center of the circle with an empty charge if it travels straight out and back. We have drawn two circles which are centered approximately at $(1, 12.5)$ km and $(5, 12)$ km. It can be observed that from the start location, the UAV cannot travel to the set of task locations approximately from $(7.5, 10)$ km to $(10, 8)$ km. However, if the UAV starts from the point $(5, 12)$ km with a full charge, it can cover those sets of task locations and return back. To enable this solution, the UAV would need to ride with the UGV along the UGV till $(5, 12)$ km, then visit the task locations within that radius and get refueled. Meanwhile, the UGV stops at $(5, 12)$ km location and waits for some time. Although, such a UGV stop helps to cover additional task locations, there are some task locations along the bottom right region that are left out. Hence, in such case, either the UGV has to have another stop along that region so that UAV can cover those task locations and utilize that UGV stop to recharge or the UGV itself should travel along that path to cover all of those task locations. From this Figure, you can see that all 3 branches intersect at a common point $(6.1, 10.8)$ km. Other scenarios are considered similar to the distribution of this scenario where three different branches meet at a point. This illustrates some of the intricacies of choosing an appropriate path for the UGV such that the UAV can successfully cover the task locations at the extreme ends. This is an optimization problem where optimal routes are

to be found for both UGV and UAV. In case of UGV, its route is modeled as a parameter set consisting of two UGV stop locations to recharge the UAVs, the wait time of UGV at those corresponding stops, and the starting or ending point of the entire route plan. The optimal solution corresponds to a UGV routes parameter set and its subjected UAV route for which the overall objective function is minimized.

In order to prove the computational efficiency, we present the results on three different scenarios shown in Figure 13. The scenarios under consideration have three branches that intersect at a single point. Each scenario has three depots. At each of the these depots, the UAV or the UGV may be recharged. The UAV may also recharged when it lands on the UGV. The UGV/UAV start their route execution from one of the three depots. This location is one of the free UGV parameter. All these scenarios consider the optimization problem for 1 UGV and 1 UAV. The UAV is a custom quadrotor with a battery capacity of 4000 mAh. The UAV and UGV velocities when moving are fixed at 10 m/s and 4 m/s respectively. The UAV and UGV fuel capacity are 287.7 kJ and 25.01 MJ respectively.

Figure 14 shows the UGV parameters for the three scenarios. The black dot on the gray rectangle represents the depot where both UGV and UAV can recharge. The large black circles represents the locations where only the UAV can recharge. Either of those depots represent the potential starting location for the UAV and UGV and is an optimization parameter. The small black circles represent the task locations that need to be visited either by the UGV or the UAV. The stopping locations for the UGV can be either in the red ellipse or the blue ellipse. In each of this ellipse, the x- and y-coordinate is a parameter (2 parameters per ellipse). For each

Parameter	Range		
	Scenario 1	Scenario 2	Scenario 3
UGV stop 1 (km,km)	(6.02,16.82) to (4.99, 11.65)	(11.36,4.86) to (14.34, 7.31)	(8.95,9.48) to (9.29, 9.38)
UGV stop 2 (km,km)	(14.70, 4.02) to (16.96,1.45)	(7.72,6.13) to (9.46, 13.02)	(8.61,9.82) to (8.61, 10.07)
UGV wait 1,2 (min)	2 to 50	2 to 50	2 to 50
Starting point	1,2, or 3	1,2, or 3	1,2, or 3

TABLE V

UGV parameters and their ranges (outer loop)

stop location, the wait time is also a free parameter (1 parameter per ellipse). The UAV/UGV may start at Depot 1, 2, or 3 (1 parameter). Table V shows the UGV parameter range for the outer level. The objective function is to minimize the time gap between completion of UAV's and UGV's routes after visiting all their task locations in the respective scenario. This kind of objective function helps to minimize the waiting time between the heterogeneous system after the route execution cycle.

In order to compare the two methods, an initial population size of 30 was chosen and both algorithms were run 3 times. Table VI compares the cost and the computational time. It can be seen that the computational time is reduced by a factor of 2 to 3 by the A-teams architecture in comparison to two-level optimization, but the cost is within 25%.

Table VII compares the A-teams solution with two-level optimization for the same initial population for the three different scenarios. From the value of the objective in the table it can

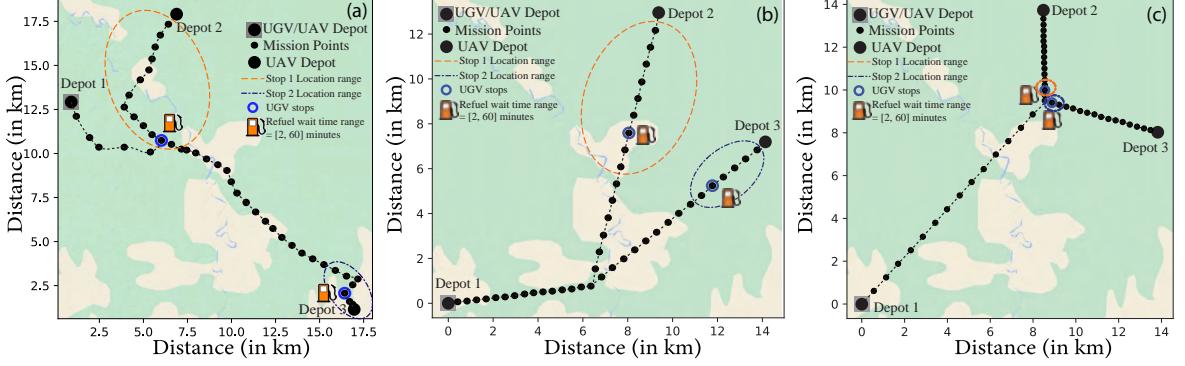


Figure 14. Description of different scenarios with parameters to be optimized. a) Scenario 1
b) Scenario 2 c) Scenario 3.

be seen that the results are mixed: A-teams is better than two-level optimization for scenario 1 but not for scenario 2 and 3. However, the difference between the two is not substantially large. The other metrics such as the UAV/UGV travel time, energy consumed, recharging stops, locations visited are also shown. There are minor differences between the two.

Figure 15 shows the final solution for scenario 1 at three different time ranges (in min): 1 – 35, 36 – 65, and 66 – 166. Since both two-level and A-teams produce very similar solution, only one solution, the two-level optimization, is shown here. The total routing time for A-teams is less than that of the two-level optimization by about 3 min. The UAV/UGV start at Depot 1 then they move together to the first stop location. Here the UAV flies to cover the locations on the top portion returning to Depot 1 to recharge. Then the UGV travels to all the task locations and returns back to the Depot 1. Figure 16 shows the coverage of task locations and the recharging stops used by the UAV-UGV for the A-teams for scenario 1. The task locations in red are those that are covered by the UAV while those in blue are the ones covered by the

Scenario type	Computational time (in minutes)		Objective (in minutes)	
	A-Teams	Two-level optimization	A-Teams	Two-level optimization
Scenario 1	37 ± 1	47 ± 2	163	166
Scenario 2	28 ± 9	82 ± 10	12	9
Scenario 3	13 ± 3	44 ± 2	18	13

TABLE VI

Comparison of total cost between A-Teams and conventional two-level optimization for different scenarios

UGV. The red cross shows the stopping locations for the UAV on the UGV for recharging. The overlaid light blue circles indicate the range of the UAV. It can be seen that the recharging stops are chosen strategically to enable maximum fuel coverage for the UAV on a single charge.

4 DISCUSSION

This paper presented the A-teams framework for optimizing the routes of a UAV-UGV pair subject to fuel and speed constraints. The A-teams framework uses asynchronous agents to create an initial pool of solutions, improve the pool, and then destroy the infeasible solutions. These agent exploit parallel architecture to produce fast solutions. When compared with conventional optimization method, A-teams produces the solution 2–3 times faster while achieving similar quality of solutions.

One advantage of A-teams is the use of asynchronous agents to improve the solutions. These agents work in parallel and hence they can be deployed independent of each other. When the

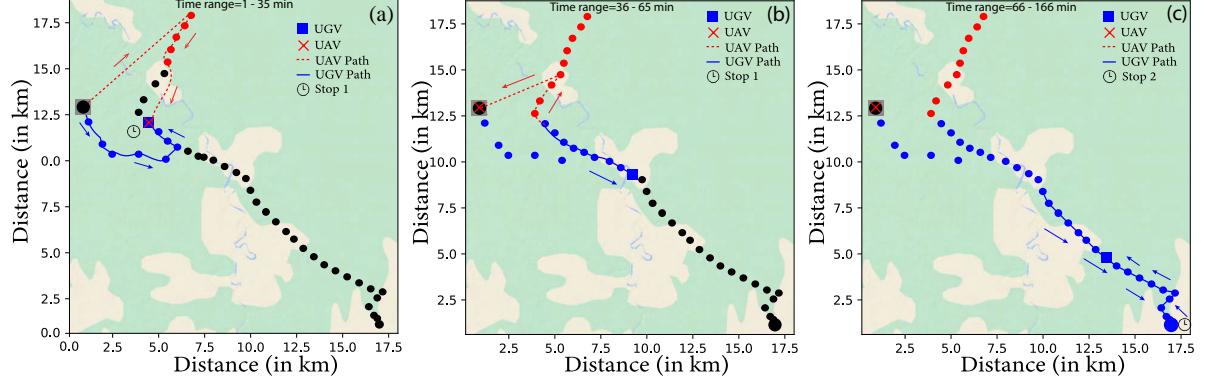


Figure 15. Solution produced by conventional two-level optimization and A-teams on Scenario 1 are indistinguishable and are shown here. The different plot shows the UAV and UGV route at various time-steps.

algorithm is deployed either on multiple core or parallel computing machines, they are able to speed up computations. Apart from that, A-Teams has the capability to scale to multi-cores where each core could be used for performing specific parallelized A-Teams computation with threads in that core. This way, a combined effort of searching for the optimal solution can be made efficiently.

Another advantage of A-teams is that the architecture seamlessly exploits the advantages of multiple algorithms to improve the solution. In our case, the genetic algorithm is used to explore the search space while the Nelder-Mead is used to locally improve the solution. Thus, we have combined a global search with local search to improve the solution quality. However, genetic algorithm is not sample efficient. One could use a sample efficient method like Bayesian Optimization if sample efficiency is important [51].

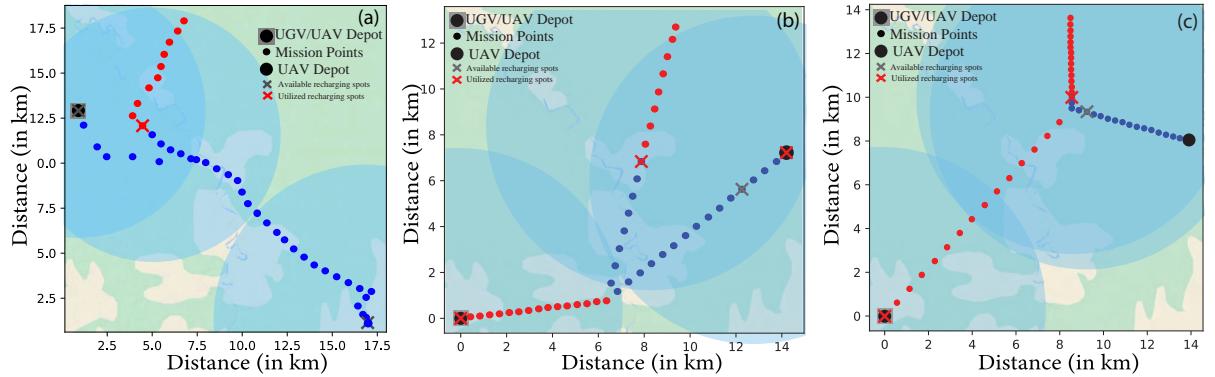


Figure 16. Optimal parameter results of respective scenarios obtained using the A-Teams architecture. (a) Scenario 1 (b) Scenario 2 (c) Scenario 3

The proposed works has some disadvantages. The UGV heuristics, the stop location and wait times, were manually determined by hand tuning. This may be overcome by using minimum set cover algorithm [61]. The quality of the initial pool of solutions created by the constructor agent is critical to ensure that the improver agent is able to improve the solution. Thus, we had to play with a few random initial guesses till we got a feasible solution as a starting base. Our results indicate that the A-teams produce superior solutions for complex scenarios (Scenario 1), but was unable to produce better solutions than our baseline method of using genetic algorithms in simple scenarios (Scenario 2 and 3) were able to. This might indicate that the more complex A-teams architecture might not be ideal for certain scenarios whose task space distribution is simple, or the number and combination of free parameters (such as stop locations, waiting times) used for the optimization problem is very large.

5 CONCLUSIONS AND FUTURE WORK

We conclude that Asynchronous multi-agent architecture (A-teams) is a competitive tool for solving the cooperative heterogeneous Vehicle Routing Problem. A-teams is able to produce good quality solutions with less computation time. It is able to do so by using specialized agents: agents to create solutions, agents to improve solutions globally and locally, and agents to destroy bad solutions.

Our future work will explore methods to automate the choosing of UGV parameters, testing the scalability of the approach by adding more UGV parameters, more UAVS and UGVs, and testing other algorithms such as Bayesian or reinforcement learning to improve the quality of the solutions as well as the solution time.

Algorithm 3 Nelder-Mead Method

Input: Current best solution of the population x_1 **Output:** Local best solution

```

11 Fix a simplex shape of  $n + 1$  vertices for  $n$  dimensions
    Set the initial simplex vertex points surrounding the current best solution
    Choose the scalar parameters  $\alpha$  (reflection),  $\beta$  (expansion),  $\gamma$  (contraction),  $\delta$  (shrink) with
    values such that  $\alpha > 0$ ,  $\beta > 1$ ,  $0 < \gamma < 1$ ,  $0 < \delta < 1$  v Evaluate the objective function value
     $f$  at each of the vertices
12 while Convergence is not achieved do
13     Sort the vertices based on  $f$  at each vertex point
        Calculate the centroid  $\bar{x}$  of the shape enclosed within the  $n$  best vertices
         $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ 
14     Perform Reflection operation to get  $x_r$ , where  $x_r = \bar{x} + \alpha(\bar{x} - x_{n+1})$ 
        if  $f(x_1) \leq f(x_r) < f(x_n)$  then
            Replace  $x_{n+1}$  with  $x_r$ 
15
16     else
17         if  $f(x_r) < f(x_1)$  then
18             Perform Expansion operation to get  $x_e$ , where  $x_e = \bar{x} + \beta(x_r - \bar{x})$ 
                if  $f(x_e) < f(x_r)$  then
                    Replace  $x_{n+1}$  with  $x_e$ 
19
20         else
21             Replace  $x_{n+1}$  with  $x_r$ 
22
23     else if  $f(x_n) \leq f(x_r) < f(x_{n+1})$  then
24         Perform Outside Contract operation to get  $x_{oc}$ , where  $x_{oc} = \bar{x} + \gamma(x_r - \bar{x})$ 
            if  $f(x_{oc}) \leq f(x_r)$  then
                Replace  $x_{n+1}$  with  $x_{oc}$ 
25
26         else
27             Perform Shrink operation
28
29     else if  $f(x_r) \geq f(x_{n+1})$  then
30         Perform Inside Contract operation to get  $x_{ic}$ , where  $x_{ic} = \bar{x} - \gamma(x_r - \bar{x})$ 
            if  $f(x_{ic}) < f(x_{n+1})$  then
                Replace  $x_{n+1}$  with  $x_{ic}$ 
31
32         else
33             Perform Shrink operation
34
35     else
36         Perform Shrink operation to get new  $x_i$ , where  $x_i = x_1 + \delta(x_i - x_1)$  for
             $i \in \{2, \dots, n + 1\}$ 

```

Algorithm 4 A-Teams Architecture

Input: Population size n , Initial population

Output: Global best solution

- 34 **Constructor Agent:** Generate random initial population with population size n
 - 35 **Constructor Agent:** Perform UAV optimization for corresponding UGV parameter set until feasibility is achieved
 - 36 **while** *Convergence is not achieved* **do**
 - 37 The following two Improver Agents work in parallel:
 - 38 **Improver Agent 1:** Perform Nelder-Mead optimization for local improvement on the current best solution
 - 39 **Improver Agent 2:** Perform Genetic Algorithm optimization for global improvement on the current population
 - 40 **Destroyer Agents 1 and 2:** Remove infeasible or duplicate solutions on the fly
 - 41 Replace initial or old population with newly generated population
 - 42 Compute the fitness value for each population member and sort them in ascending order
-

Parameter	Optimal parameter values					
	Scenario 1		Scenario 2		Scenario 3	
	A-Teams	Two-level	A-Teams	Two-level	A-Teams	Two-level
UGV stop 1 location (km,km)	(4.99,11.65)	(4.99,11.65)	(7.92,6.90)	(11.36,4.86)	(8.61,10.08)	(8.95,9.48)
UGV stop 2 location (km,km)	(16.96,1.45)	(16.96,1.45)	(12.36,5.68)	(8.30,8.43)	(9.29,9.38)	(8.61,10.08)
UGV stop 1 wait time (min)	20	20	50	21	20	22
UGV stop 2 wait time (min)	20	21	20	21	20	20
Route starting and ending point	Depot 1	Depot 1	Depot 3	Depot 3	Depot 3	Depot 2
Metrics		Scenario 1		Scenario 2		Scenario 3
		A-Teams	Two-level	A-Teams	Two-level	A-Teams
Objective function (min)	163	166	12	9	18	13
Total time (min)	228	231	216	201	145	131
UGV results						
Travel time (minutes)	228	231	216	201	145	131
Energy consumed (MJ)	23.16	23.19	19.50	20.89	8.16	6.31
# Locations visited	34	34	23	25	17	17
UAV results						
Travel time (minutes)	65	65	204	192	127	118
Energy consumed (kJ)	460.65	460.65	1082.92	1301.78	841.72	874.73
Recharging stops on UGV	1	1	2	3	2	2
Recharging stops on Depot	0	0	2	2	1	1
# Locations visited	10	10	23	21	29	29

TABLE VII. Comparison between A-Teams and conventional two-level optimization on metrics from the optimal solution for different scenarios. These results are shown for a specific initialization of population by Constructor agent.

CHAPTER 5

COMPUTATIONALLY EFFICIENT MULTI-AGENT OPTIMIZATION FRAMEWORK FOR ONLINE ROUTING OF UAV-UGV SYSTEM: VARIANT OF A-TEAMS

Overview: Unmanned Aerial Vehicles (UAVs) have the ability to monitor vast areas but are limited in their battery capacity. The collaboration with Unmanned Ground Vehicles (UGVs) can significantly enhance the endurance and potential of UAVs by utilizing them as mobile recharging vehicles. However, such collaboration increases the complexity of planning the routes of the vehicles. For practical applications, it's crucial to efficiently develop high-quality UGV-UAV routing solutions within a reasonable time frame and quickly adapt those routes to changing conditions, if any. In this paper, we propose an improved method for multi-agent optimization framework that provides computationally efficient online UGV-UAV route solutions. The effectiveness of the optimization framework is validated with several scenarios by comparing it across standard meta-heuristic baselines like the Genetic Algorithm in simulation. The proposed framework produces near-optimal solutions that are computed 40% faster than the baseline while maintaining the solution quality within 2%. Additionally, the framework's com-

Parts of this chapter is taken from the followign published journal article:
Ramasamy, S., Mondal, M. S., Humann, J. D., Dotterweich, J. M., Reddinger, J. P. F., Childers, M. A., & Bhounsule, P. A. (2024, August). 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) (pp. 204-211). IEEE.

putational efficiency is validated using hardware (<http://tiny.cc/ngjkxz>), demonstrating its ability to do online planning when mission points are dynamically added or removed from the scenario.

Keywords: Write keywords here

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have rapidly evolved across diverse fields such as entertainment, logistics, surveillance, and disaster relief management due to their small size, relatively low cost, agility, and ease of usage [62]. However, the major limitation of using such UAVs is their restricted operational time due to limited battery capacity, which does not allow them to perform critical tasks such as surveillance and disaster relief that either demand large-scale usage or longer duration.

To perform tasks of high endurance over wider areas, UAVs could be paired up with Unmanned Ground Vehicles (UGVs) to provide them with mobile recharging platforms [63]. In this setup, due to their relatively fast speeds and high altitude, UAVs can provide a bird's eye view of the ground and return to UGVs for recharging before taking off again. The collaborative routing of Unmanned Aerial and Unmanned Ground Vehicle Systems is an NP-Hard combinatorial optimization problem [26]. The traditional route planning algorithms often struggle with online planning due to the challenges of combinatorial optimization. Our approach overcomes these issues by developing an optimization framework that can provide online planning for UAV-UGV collaborative routing.

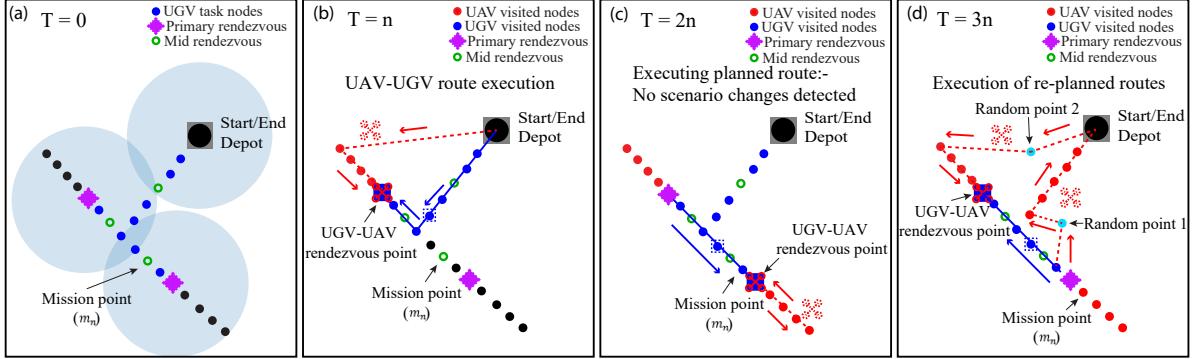


Figure 17. Persistent surveillance illustration for a collaborative UGV-UAV routing problem. For a single Persistent Surveillance mission, steps a) through d) occur sequentially, where 'n' represents the frame rate. a) A candidate UGV-UAV task visit points. b), c), d) Execution of the planned/re-planned route. Steps (c) and (d) are repeated for persistent node visits. Re-planning is done online if dynamic changes are observed.

2 RELATED WORK

Several works in the literature have addressed the cooperative UGV-UAV routing problem. Liu et. al. [64] solved a cooperative Ground Vehicle(GV)-UAV routing problem for surveillance and reconnaissance missions. They proposed a method that first created a large UAV route without considering battery limits. Then, they use a Split Heuristic to split this route into feasible parts connected by GVs to complete the route. Gao et. al. [46] formulated a cooperative UGV-UAV routing problem for emergency resource delivery during COVID-19, using a system that accepts operation orders through an intelligent module and plans routes with a Mixed Integer Linear Programming (MILP) method, followed by an iterative improvement algorithm. Seyed et. al. [65] addressed the problem of persistent surveillance by using energy-constrained

UAVs and the UGVs. The UGVs acted as mobile recharging stations to recharge UAVs. The UGVs and UAVs were organized as a set of UGV-UAV teams. The environment was optimally partitioned into several segments, where each team performed persistent surveillance of task points in those respective partitions in a cyclic fashion. The task points of each partition were visited optimally by the UGV-UAV team using the Dantzig-Fulkerson-Johnson (DFJ) linear programming algorithm. These studies are limited in utilizing specific optimization algorithms for UGV and UAV routing, missing out on the potential benefits of combining different algorithms. The combination could enhance solution quality and computational efficiency by leveraging the strengths of each algorithm together. [66].

The proposed research work deals with improving a multi-agent optimization framework called Asynchronous Teams (A-Teams) [53]. 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. [67] 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. [68] introduced MultiAGent Metaheuristic Architecture (MAGMA), an optimization framework that facilitated cooperation between Memetic algorithms and GRASP (Greedy Randomized Adaptive Search Procedures). In MAGMA, agents were operated at vari-

ous 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.

On the experimental front, limited work has been done to implement the different routing and path planning algorithms on the UAV-UGV hardware system. Nigam et al. [69] solved the persistent surveillance problem of multiple UAVs by developing a optimal control policy structure to navigate on a gridded target space and validated it on their hardware testbed for up to 4 UAVs. The policy dictates the UAVs' action to move front, back, or side based on the age period of each grid cell. Karapetyan et al. [70] demonstrated an approach to solve a coverage path planning problem for a UGV-UAV system using a two-step algorithm. Their two-step algorithm first ignored UAV energy limits to plan the overall coverage path. In the first step, both UGV and UAV optimal coverage trajectories across an area are planned using the Boustrophedon Cellular Decomposition (BCD) algorithm. Then, they divided the area into clusters considering the UAV energy limits, using bipartite graph matching to link UAV and UGV clusters efficiently. They tested their approach with an outdoor UAV-UGV system to prove its effectiveness.

Some works in the literature have also considered re-planning UAV-UGV paths owing to dynamic changes in the environment. Ni et al. [71] introduced a 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. [72] 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. [73] tackled a real-time navigation problem for urban firefighting with UAVs and UGVs in GNSS-denied areas. They used Monte-Carlo localization to accurately determine the robots' positions in a pre-mapped environment, a Lazy Theta* algorithm for global path planning, and 3D-LIDAR mapping to enable the local path planner to quickly adjust plans 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.

In this paper, we propose an enhanced multi-agent optimization framework (A-Teams) with a new **Predictor Agent** for achieving computationally efficient, near-optimal solutions. The Predictor Agent predicts the feasibility of UGV route parameters, reducing unnecessary computations. We validated the framework on actual hardware, showcasing online planning amid dynamic changes. Our novel contributions are: 1) A computationally efficient framework for online heterogeneous UGV-UAV routing. 2) Implementation and testing of autonomous online route re-planning on hardware under dynamic conditions. 3) A faster approach for the Dynamic Vehicle Routing Problem (DVRP) applied to a heterogeneous UGV-UAV system. This approach stands out because it shifts away from the conventional application of DVRP, which primarily concentrates on logistic trucks and vehicles.

3 METHODS

3.1 Problem formulation

To address the collaborative UGV-UAV routing problem for remote scenarios that are less accessible to humans, we define a set of task points, $\mathcal{M} = \{m_0, m_1, \dots, m_n\}$, located within a

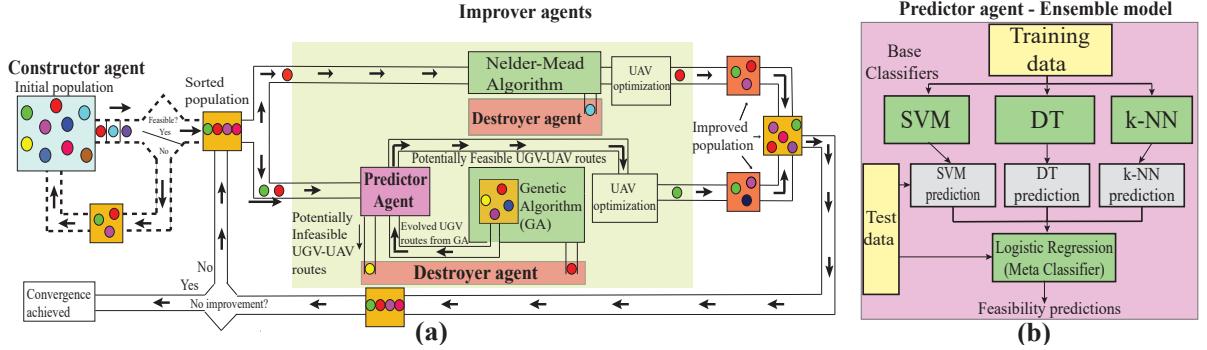


Figure 18. Description of the proposed A-Teams optimization framework used for UGV-UAV routing. a) This proposed A-Teams has Constructor, Improver, Destroyer and Predictor agents. Agents strategically choose UGV routes, which are fed into UAV optimization to get collaborative UGV-UAV route outputs. b) Training of Ensemble model in Predictor Agent

guided road network. The UGV and UAV start from the depot D . These points are considered for persistent surveillance by both UAV and UGV. The UAV can either get recharged on UGV or at D . The vehicle system is of a heterogeneous nature where the UAV travels with higher velocity v_a but is limited in its energy capacity. Thus, it can be paired with a slower-moving UGV with velocity v_g , which has a larger energy capacity E_g to recharge the UAV and power its components. The operational range of the UAV with full energy E_a is visualized as blue circles in Figure 17 a), demonstrating that a single UAV cannot adequately cover all task points, thus substantiating the need for strategic UGV routing to ensure full UAV task coverage and efficient rendezvous. This interdependence between UAV and UGV routes necessitates a bi-level optimization approach, where the UGV route is optimized at the outer level, and the UAV route is optimized at the inner level. This is based on the idea of “UGV first, UAV second” approach.

The optimization of the UGV route is formulated as a set of free parameters, denoting the rendezvous locations that the UAV makes with the UGV. The complete parameter set for a candidate UGV route, X_s , is made up of two subsets: primary rendezvous locations $X_p = \{x_1, x_2, \dots, x_j\}$ aimed at recharging as well as directing the UAV to new task regions, and mid-rendezvous locations $X_m = \{x_1, x_2, \dots, x_k\}$ for UAV recharging in middle of the UGV's travel. To identify primary rendezvous points, the Minimum Set Cover (MSC) Problem is solved using Constrained Programming, as detailed by Mondal et al. [74]. This approach tackles the NP-Hard nature of the MSC problem, allowing for multiple X_p sets depending on the scenario's scale and complexity. Equations Equation 3.1-Equation 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 time of flight T_f^a and velocity v_a of the UAV. Mid-rendezvous points (X_m), used for UAV recharging, 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, the inner-level UAV optimization is solved for near-optimal UAV routes. The UAV optimization is formulated as an Energy-constrained Vehicle Routing Problem (E-VRP) subjected to satisfying UGV route and UAV fuel constraints. The UAV route is given as feedback to the outer level for UGV optimization. The feedback indicates whether the UAV provides feasible or infeasible results for a UGV route. The feasibility criteria consider that the UAV should visit all its task points at least once.

$$\min \quad j \quad (3.1)$$

s.t.,

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

where

$$j = |X_p|, k = |X_m| = j + 1, R = 0.5 \cdot T_f^a \cdot v_a \quad (3.3)$$

The overall objective of this problem is minimizing the total time T to perform persistent surveillance of UGV-UAV task points subjected to dynamic changes until the UGV runs out of fuel. For instance, Figure 17 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 $\mathcal{R} = \{r_0, r_1, \dots, r_n\}$. 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). Irrespective of those changes, Figures c) and d) happen in a loop until the UGV runs out of fuel. Due to the specificity of the problem nature

and the scenarios considered, the persistent surveillance is ensured by adding a penalty P_t of at least one task point in \mathcal{M} to be visited more than once to the objective function. If all points are visited only once or some are not visited, a large penalty $P_t = L$ is added to the objective. This discourages infeasible or one-time surveillance solutions and encourages solutions with persistent surveillance of the task points. The objective function for the UGV-UAV routing is denoted mathematically as follows. Here $|v(m_n)|$ denotes the number of visits of a point m_n where, $m_n \in \mathcal{M}$. Once the near-optimal UGV-UAV routes are obtained, the set \mathcal{M} gets divided into \mathcal{M}_g for UGV visits, and \mathcal{M}_a for UAV visits, such that $\mathcal{M}_g \cup \mathcal{M}_a = \mathcal{M}$, thus allocating distinct tasks between UGV and UAV.

$$\min \quad T + P_t \quad (3.4)$$

where,

$$P = \begin{cases} 0, & \text{if } |v(m_n)| > 1 \text{ for atleast one } m_n, \\ L, & \text{otherwise, where } L \text{ is a large number} \end{cases} \quad (3.5)$$

The different optimization levels for UGV and UAV routes are described in the following subsections.

3.2 Proposed optimization framework

3.2.1 Solving outer-level UGV routing using Asynchronous Teams (A-Teams) framework

Figure 18 a) represents the proposed framework to perform UGV-UAV optimization. A-Teams is a multi-agent framework that uses a team of algorithms to optimize a given problem. The agents in the framework possess distinct functionalities and solve a problem through cooperation and achieve better solutions than their individual counterparts. There are four agents: **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 the existing A-Teams framework can be found in [75].

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 $\mathcal{N} = 40$. These routes are sent to the inner-level block for UAV optimization 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 18 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 visited at least once. The Ensemble model, tested with UGV routes generated by a Genetic Algorithm, filters out infeasible routes before UAV optimization, thereby enhancing process efficiency.

3.2.2 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 fuel 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 $V = \{0, 1, 2, \dots, m, D\}$ 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 UAV could utilize to get recharged. Let c_{ij} be the non-negative arc cost between a particular i and j . Let t_{ij} be the time travel 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. We formulate the VRP problem with fuel constraints, time windows, and dropped visits. The mathematical formulation for EVRP is presented here, but more details may be found in [51].

Objective:

$$\min \sum_{i \in V} \sum_{j \in V} t_{ij} x_{ij} \quad (3.6)$$

Major constraints:

$$f_j \leq f_i - (P^a t_{ij} x_{ij}) + L_1(1 - x_{ij}), \forall i \in V, j \in V \setminus D \quad (3.7)$$

$$f_j = Q, \quad \forall j \in D \quad (3.8)$$

$$0 \leq f_j \leq Q, \quad \forall j \in V \quad (3.9)$$

$$t_j \geq t_i + ((t_{ij} x_{ij})) - L_2(1 - x_{ij}), \forall i \in V, j \in V \quad (3.10)$$

$$t_j^l \leq t_j \leq t_j^u, \quad \forall j \in V \quad (3.11)$$

$$x_{ij} = 1 \rightarrow \sum_{i \in V \setminus D} x_{ji} = 1, \forall j \in D, \forall i \in V \setminus D \quad (3.12)$$

The objective is Eq. Equation 3.10 is to minimize the time to complete UAV routing. The constraint in Eq. Equation 3.11 is the Miller-Tucker-Zemlin (MTZ) formulation [18] for sub-tour elimination which enables that none of the UAVs are fully drained out while eliminating loops. P^a in this equation represents the UAV power consumption curve, which will be provided in Section 4. Constraint Eq. Equation 3.12 states that if the vertex is a recharging UGV

stop, UGV must refuel the UAV to its full capacity Q . Constraint Eq. Equation 3.13 is the condition that the UAV's fuel at any vertex in V should be between 0 and maximum fuel capacity. Constraint Eq. Equation 3.14 denotes that the cumulative arrival time at j^{th} node is equal to the sum of cumulative time at the node i , t_i and the travel time between nodes i and j , $t_{ij}x_{ij}$. Constraint Eq. Equation 3.15 is the time window constraint that tells the vehicle to visit a certain vertex in the specified time window for that node. The constraint in Eq. Equation 3.16 ensures that if any UAV comes to the refuel vertex to recharge, an arc must exist between that refuel 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. [17] conducted research that tested the 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. More details about this can be found in [17].

3.3 Hardware setup

Figure 19 displays the lab-based architecture used to perform the hardware experiment. We utilized a DJI Tello drone (80g) and a HiWonder MasterPi robot, controlled via a 2.4GHz WiFi connection and socket communication, respectively. The central manager, integral to the hardware framework, uses A-Teams to determine optimized waypoints for UGV-UAV op-

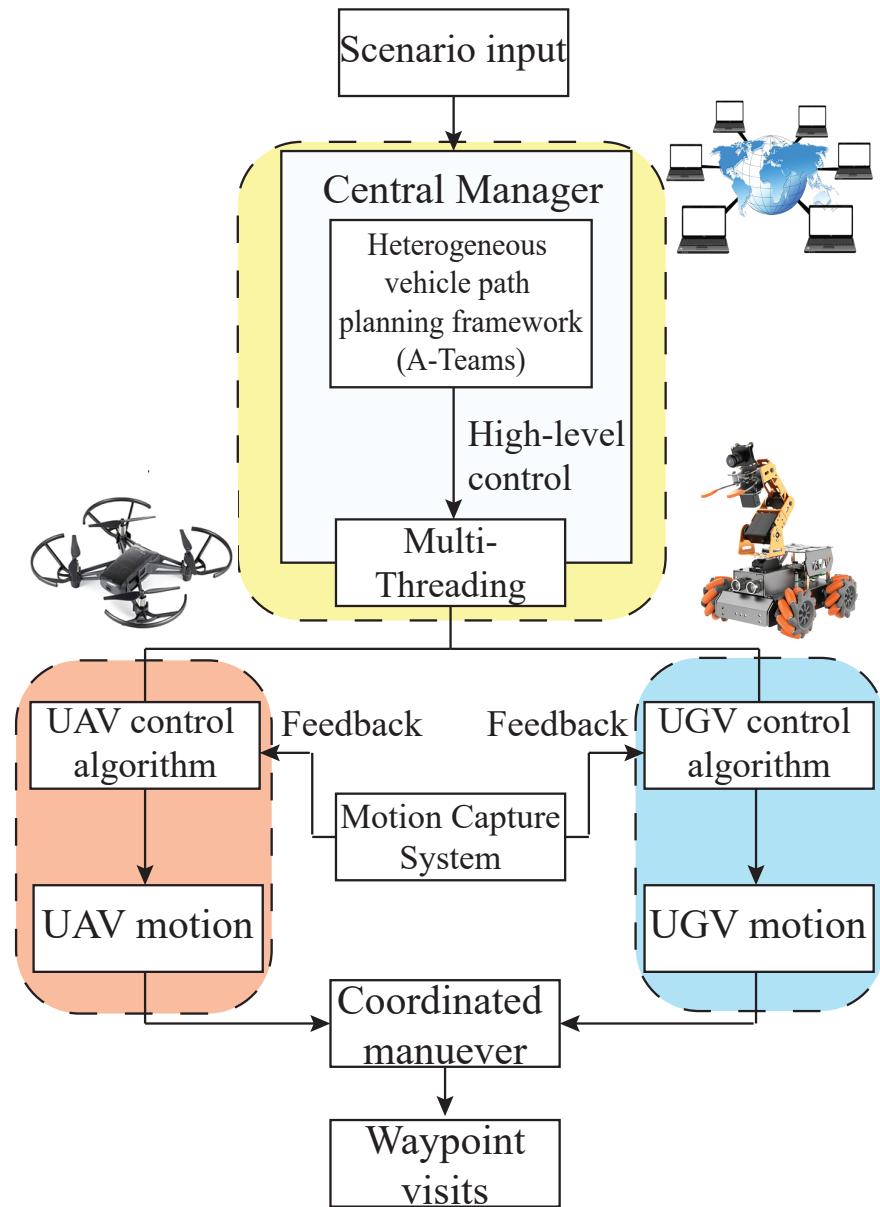


Figure 19. Hardware experimental framework

erations. It generates route sequences in the form of YAML files for each vehicle, transmits pre-planned routes in static scenarios, and adapts with re-planning for dynamic changes. This system allows for parallel processing and online communication with the vehicles. Additionally, a motion capture system monitors the vehicles' trajectories, providing feedback to ensure adherence to the designated paths and coordinating UAV rendezvous with the UGV.

4 RESULTS

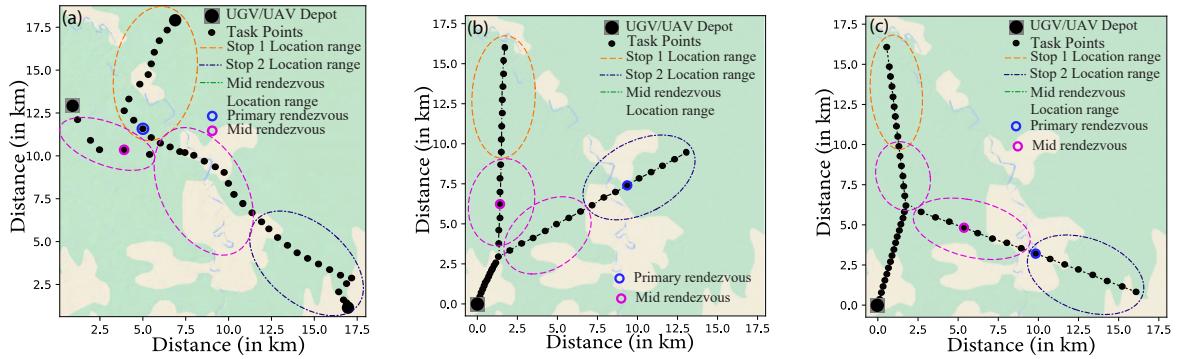


Figure 20. Scenario descriptions containing the UGV free route parameters for the optimization process. Primary rendezvous location parameter set (orange and blue ellipse) are solved from Minimum Set Cover problem a) Scenario 1 b) Scenario 2 c) Scenario 3

4.1 Evaluation of the proposed framework

The proposed framework is evaluated on different scenarios to test its generalizability, followed by hardware implementation on one scenario. We used Python 3 for all the computations: the Ensemble of different classification algorithms (kNN, SVM, Decision Trees) used

in the Predictor agent is from the Scikit-learn package, a custom-written Genetic Algorithm, and Nelder-Mead from the Scipy package for performing UGV free parameter optimization; and OR-Tools 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.

Figure 20 depicts three distinct scenarios for simulation, designed to facilitate comprehensive computational analysis through varied spatial settings. To ensure robustness, the initial population for simulation is randomized multiple times for each scenario. The scenarios consider a single UAV and UGV, with the UAV having a battery capacity of 4000 mAh (total energy of $E_a = 287.7\text{kJ}$) and the UGV having an energy capacity of $E_g = 25.01\text{MJ}$. The UAV and UGV are set to travel at velocities of $v_a = 10\text{m/s}$ and $v_g = 4.5\text{m/s}$, 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 $P_g = 464.8v_g + 356.3$ (referred from [76]). For the considered scenarios, a candidate UGV route X_s has two primary rendezvous locations ($j = 2$) and three mid UGV-UAV rendezvous locations ($k = 3$), adding up to a total of 5 parameters ($S=5$) for the UGV route.

The proposed method is compared against existing methods including the standard A-Teams framework without a Predictor agent and a parallelized version of the traditional GA, with GA's stopping condition set at either population convergence or a maximum of $G= 20$ generations. For A-Teams as well as GA population initialization, the sample size for performing UGV free parameter optimization is considered to be $N= 40$, according to [77]. Table VIII shows the optimization metrics and Table IX shows the Predictor agent's prediction quality. The simulation results in Table VIII indicate that A-Teams equipped with the Predictor agent

TABLE VIII

Optimized solution and computational comparison between proposed framework and standard meta-heuristics

Scenario type	A-Teams with Predictor agent for UGV routing & CP for UAV routing		A-Teams without Predictor agent for UGV routing & CP for UAV routing		Genetic Algorithm for UGV routing & CP for UAV routing	
	Objective value (min.)	Computing Time (min.)	Objective value (min.)	Computing Time (min.)	Objective value (min.)	Computing Time (min.)
Scenario 1	212.3 ± 3.8	9 ± 0.6	210.8 ± 3.5	13 ± 0.5	212.3 ± 3.8	47 ± 4.4
Scenario 2	233.3 ± 6.9	8 ± 1.9	233.8 ± 7.2	12 ± 1	228 ± 8.1	39 ± 12
Scenario 3	191 ± 7.8	4 ± 1	190.6 ± 8.1	7 ± 1.5	190 ± 8.4	45 ± 7.5

achieve objective results that are comparable to other methods across scenarios, while reducing computational time up to 30% compared to conventional A-Teams and by up to 70% compared to the GA-only meta-heuristics. This demonstrates the framework's generalizability and computational efficiency without sacrificing solution quality compared to standard meta-heuristics.

Table IX evaluates the prediction quality between two Machine Learning models (proposed Ensemble vs simpler k-NN model) used in the Predictor agent by comparing them across four metrics. Eq. Equation 4.1 outlines those metrics. In the equations, **True Feasible (TF)** represents the number of accurately predicted feasible UGV routes, while **True Infeasible (TI)** correctly identifies the number of infeasible ones. Conversely, **False Infeasible (FI)** occurs when feasible routes are mistakenly labeled infeasible, and **False Feasible (FF)** happens when infeasible routes are wrongly labeled feasible. The table reveals that the Ensemble model outperforms k-NN across all the metrics considered, demonstrating the Predictor agent's capability to differentiate feasible from infeasible UGV-UAV routes accurately.

TABLE IX

Predictor agent: Assessment of classification performance with proposed Ensemble vs simpler k-NN model

Scenario #	Accuracy (%)		Precision (%)		Recall (%)		F-score (%)	
	Ensemble	k-NN	Ensemble	k-NN	Ensemble	k-NN	Ensemble	k-NN
Scenario 1	99.3 ± 0.6	99.3 ± 0.6	100 ± 0	100 ± 0	92 ± 6.9	92 ± 6.9	95 ± 4	95 ± 4
Scenario 2	91 ± 11.4	81 ± 1.53	85 ± 26.5	68.3 ± 11	94 ± 7.2	90 ± 7.4	86.6 ± 15	73.3 ± 2.1
Scenario 3	94.3 ± 2.6	90 ± 5.2	83 ± 11.8	80 ± 14.3	89.8 ± 7.2	73.5 ± 12.5	85.5 ± 11.5	76 ± 8.1

Figure 21 presents a 3D plot of three UGV route parameters, illustrating the classification comparison between the Ensemble model and the k-NN model for Scenario 2. All coordinate values are in kilometers. In this plot, green solid dots represent feasible UGV-UAV route solutions, while red dots indicate infeasible ones. Comparing the two plots reveals that the Ensemble model achieves better classification performance than the k-NN model, as the k-NN model struggles to clearly distinguish the boundary between feasible and infeasible solutions.

$$\begin{aligned}
 \text{Accuracy} &= \frac{TF + TI}{TF + TI + FF + FI} (\%) \\
 \text{Precision} &= \frac{TF}{TF + FF} (\%) \\
 \text{Recall} &= \frac{TF}{TF + FI} (\%) \\
 F\text{-score} &= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} (\%)
 \end{aligned} \tag{4.1}$$

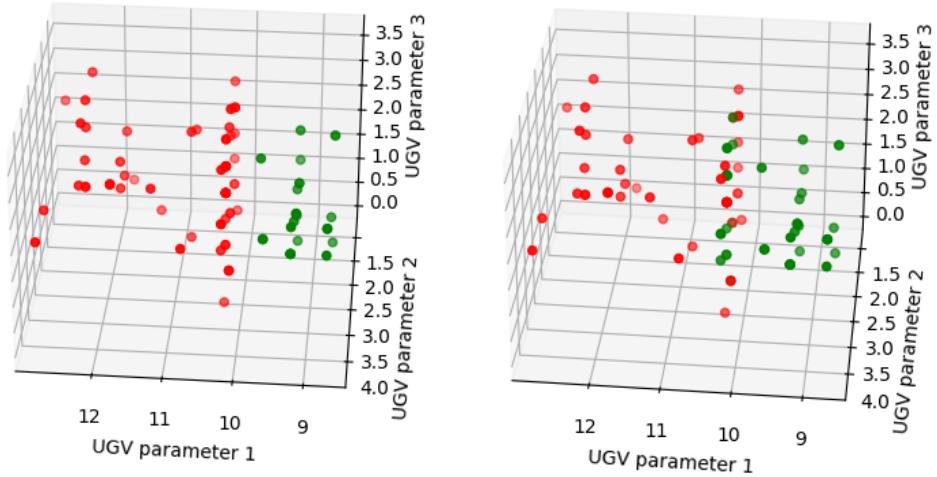


Figure 21. Comparison of the Predictions made by Ensemble model vs kNN model for Scenario 2. Left - Ensemble Model; Right - kNN Model

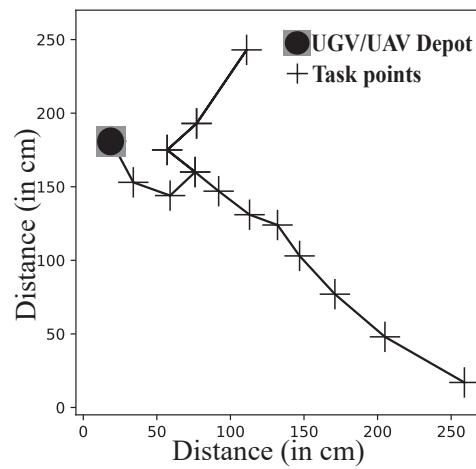


Figure 22. 2D plot of experimental scenario

4.2 Hardware re-planning with dynamic changes

Figure 22 shows the scaled scenario considered for performing the hardware experiments with real-time dynamic changes. The chosen scenario for the hardware is scaled down to a 250×250 cm area in the lab. The experiments took place in the well-equipped Robotics and Motion Laboratory at the University of Illinois Chicago, which includes a high-fidelity motion capture system. To perform the route optimization, the scenario is scaled up to match the required UAV-UGV power consumption and recharging specifications mentioned in subsection 4.1, as those equations pertain to the large-scale scenarios seen in Figure 20. Once the optimized route plan is obtained, the UAV and UGV task point visit sequence is scaled down accordingly. The vehicle speed for UAV hardware is considered to be 0.15 m/s, and for UGV it is 0.4 m/s. When the UAV lands on UGV, it waits for ‘X’ seconds to model the recharging time.

Multiple trials of the experiment were conducted. The central manager uses the proposed framework to perform offline route planning in a scenario with no dynamic changes. Once the pre-planned routes are obtained for the system, those routes are fed as commands to the UAV and UGV respectively. For the scenario with dynamic changes, a few random dynamic targets would appear during the middle of the route execution, and the central manager would re-plan the UGV and UAV routes to accommodate those changes. Figure 23 shows the dynamic UAV targets appearing randomly whenever a rendezvous between UAV and UGV is about to happen. Once the re-planning is done, the updated route commands are fed to the vehicle system. The experiment takes only about **1.5 minutes** to provide optimized routes for a planning horizon of **8 minutes**. More details can be seen from the experimental video

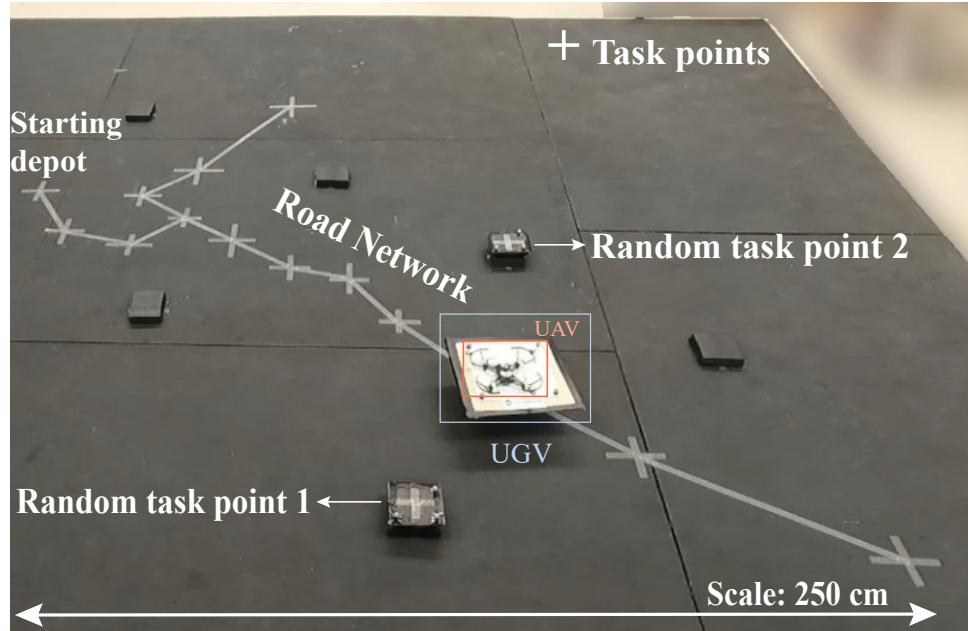


Figure 23. Experiment instance with dynamic changes

here: <http://tiny.cc/ngjkxz>. Since the proposed framework delivers near-optimal solutions quickly, the process of re-planning happens online.

In the lab, we scaled down the UAV's recharge and flight times by a factor of 60 (e.g., recharging time was scaled from 15 minutes to 15 seconds). Also, a buffer time is added to account for the drone's takeoff and landing time in the simulation. After multiple trials, the buffer time to be added in simulation results for takeoff and landing was obtained at 10 and 15 seconds on average. Thus, the maximum flight time for a single charge is replicated to be 50 seconds, denoted as the Maximum endurance limit for a single flight. The time of flight is scaled

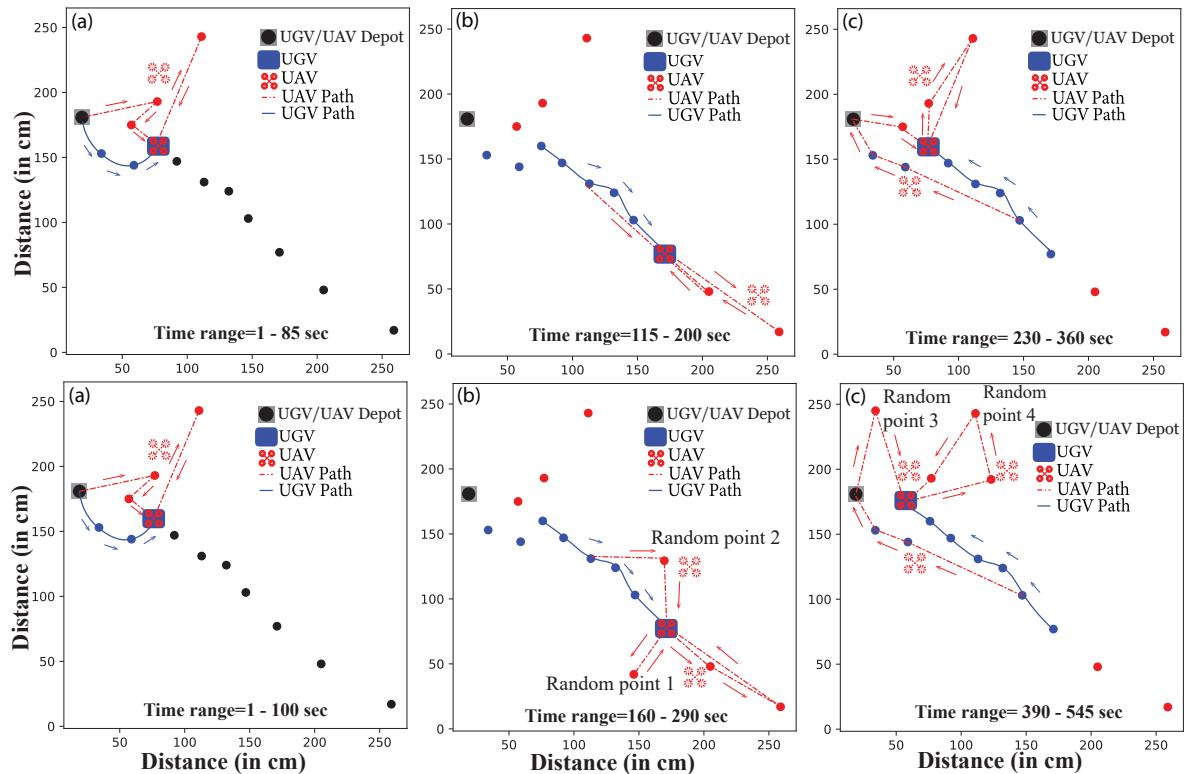


Figure 24. UAV-UGV simulated route plan obtained from the multi-agent optimization framework (A-Teams). Top Row shows the initial planned route without any dynamic changes. Bottom Row shows re-planned routes considering the dynamic changes in the scenario. Animation of simulated routes can be viewed here: <http://tiny.cc/mgjkxz>

down from 25 minutes to 25 seconds and an additional time of 25 seconds to accommodate the buffer time for takeoff and landing.

Figure 24 illustrates the simulated route patterns for the case study scenario. The top row shows the static scenario without dynamic changes, where the UAV and UGV follow pre-planned routes. The bottom row depicts adaptive routing under dynamic conditions, with updated route commands adjusting the patterns when changes are detected. For instance, seeing the Figure 24 c) of the top and bottom row, the UAV’s route includes additional task points, and the UGV’s route is updated accordingly to meet the objective of the problem. These results are validated with hardware experiments, and Figure 25 compares UAV flight and rest durations between simulation and actual hardware. Discrepancies in the figure are noted due to hardware uncertainties. The positive slope in the figure represents UAV takeoff and flight phases, while the negative slope indicates the recharging phase. The figure demonstrates that actual landing and takeoff times may differ from simulations, contributing to overall discrepancies.

5 DISCUSSION

This research work develops a computationally efficient optimization framework that has the ability to plan the cooperative UGV-UAV routing online. The key to computational efficiency comes from the algorithms used in the proposed A-Teams framework, where the Predictor agent used an Ensemble Machine Learning model to discard infeasible route solutions. Compared to conventional A-Teams, the proposed framework improves computational efficiency by up to 36% while maintaining solution quality within 1%. Against the Genetic Algorithm, it achieves up to 78% efficiency with solution quality within 2%.

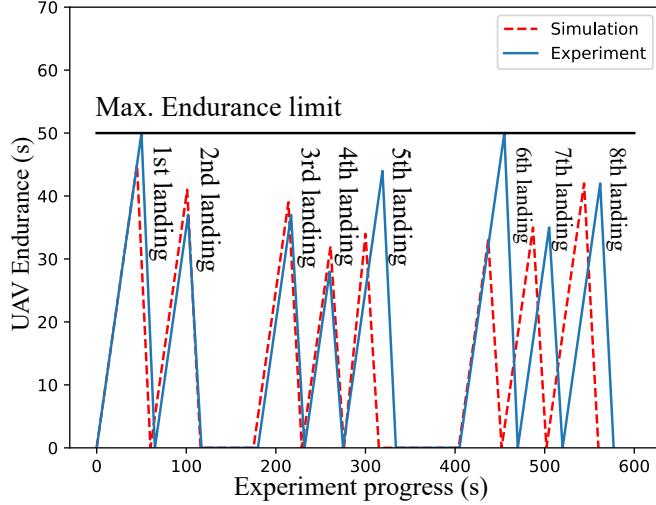


Figure 25. Hardware vs simulation Time-of-Flight comparison of lab setup experiment

The A-Teams multi-agent framework enhances solutions by integrating different algorithms, each with unique strengths. For instance, pairing a global search like Genetic Algorithm with a local search such as Nelder-Mead, streamlines the search and quickly yields near-optimal solutions, unlike the GA-only method. Also, CP for UAV routing leads to inexpensive optimization using heuristics, which helps achieve an efficient computation. For the Predictor agent, an Ensemble model that integrates several base classifiers such as k-NN, SVM, and Decision trees is chosen for making a robust prediction about discarding the UGV-UAV route solutions that are potentially infeasible. This is because one base classifier might give a contradicting prediction compared to the other. Hence, the Ensemble meta-classifier tries to make a balanced prediction with a more accurate output. In Table IX both the Ensemble and k-NN classifiers give the same prediction results for Scenario 1. This is partly due to the distinct spatial distribution

of task points in Scenario 1, with drastically different branch lengths connecting the midpoint. This distribution made it easier for the classifiers to discern between feasible and infeasible solutions, resulting in identical predictions.

The proposed work has some limitations. In the context of the optimization framework, the Predictor agent's quality depends upon the base classifiers used, which depend on several hyperparameters. Currently, the Predictor agent uses default hyperparameter settings from Scikit-Learn, such as the 'number of neighbors' parameter N in k-NN set to 5, SVM kernel set to Radial Basis Function (RBF), and 'gini' index for tree splitting in Decision Trees. Poor Predictor Agent may discard good solutions and result in an optimal solution worse than conventional A-Teams. Future work would address this by performing hyperparameter optimization to have the classifiers tailored to this UGV-UAV prediction and thus have a better predictive model. The closeness of hardware results to simulation depends upon the type of hardware used. Since we performed the experiments with a scaled hardware setup in the lab, there are considerable discrepancies between the simulation and hardware results regarding delays in takeoff/landing times of the UAV from/on the UGV. Future efforts will focus on improved hardware, developing a simulation pipeline between routing and hardware deployment, and outdoor testing where there is a large variability compared to indoor testing.

6 CONCLUSION

In this study, a computationally efficient A-Teams framework integrated with a Predictor Agent has been developed. The developed framework is demonstrated to generate near-optimal

solutions faster than the existing methods. Furthermore, it was validated in actual setting by conducting a hardware experiment on a case study scenario subjected to dynamic changes.

Our future work will focus on improving the Predictor agent’s ensemble model through hyperparameter optimization. Additionally, we will explore alternative objective functions for persistent surveillance, such as minimizing the maximum age period, and extend these improvements to outdoor experiments.

CHAPTER 6

REINFORCEMENT LEARNING ASSISTED A-TEAMS FOR ADAPTIVE ALGORITHM SELECTION IN UAV-UGV OPTIMIZATION PROCESS

Overview: 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, the increase in their potential also increases the complexity of routing these heterogeneous vehicles. Therefore, developing an efficient UGV-UAV routing framework is essential to ensure real-time adaptability in this autonomous system, enabling it to respond effectively to dynamic changes. An end-to-end autonomous approach capable of making its own decisions during the optimization process can significantly enhance computational efficiency. In this paper, we introduce a novel learning-based hyperheuristic approach using Reinforcement Learning (RL) within a multi-agent optimization framework (A-Teams) to deliver computationally efficient, real-time UGV-UAV routing solutions. 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) an A-Teams variant, (2) conventional A-Teams, and (3) Genetic Algorithm. The proposed framework is tested on single UAV-UGV and multi UAV-UGV systems and 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.

Parts of this chapter is taken from published journal article

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 [62]. 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 [63]. In this cooperative setup, UAVs can leverage their speed and altitude to provide a comprehensive aerial view, returning to UGVs for recharging as needed. The collaborative routing of UAV-UGV systems, however, presents an NP-Hard combinatorial optimization challenge [26]. 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, 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. [64] addressed a cooperative Ground Vehicle (GV)-UAV routing problem for surveillance and reconnaissance missions. Their approach initially generates an extensive UAV route

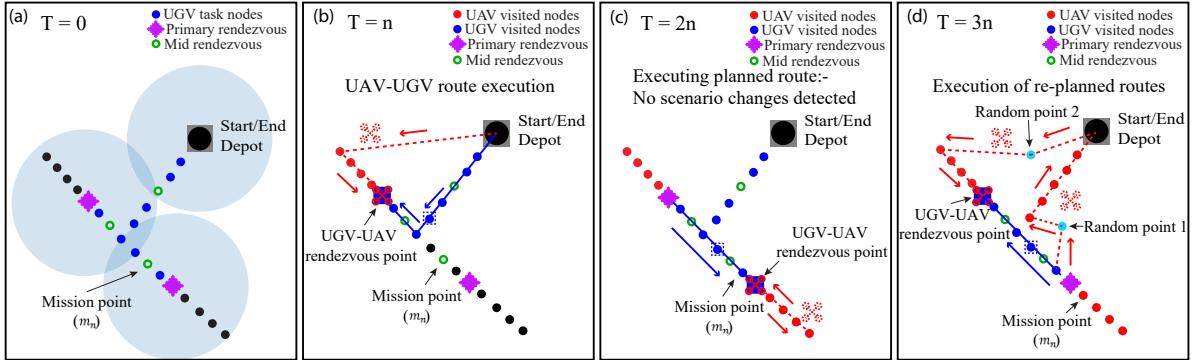


Figure 26. Persistent surveillance illustration for a collaborative UGV-UAV routing problem. For a single Persistent Surveillance mission, steps a) through d) occur sequentially, where 'n' represents the frame rate. a) A candidate UGV-UAV task visit points. b), c), d) Execution of the planned/re-planned route. Steps (c) and (d) are repeated for persistent node visits.

Re-planning is done online if dynamic changes are observed. 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. [46] formulated a cooperative UGV-UAV routing problem for emergency resource delivery during COVID-19. Their system processes operation orders via an intelligent module, plans routes using a Mixed Integer Linear Programming (MILP) approach, and refines these routes through an iterative improvement algorithm. Seyedi et. al. [65] addressed the problem of persistent surveillance using energy-constrained UAVs supported by UGVs, which served as mobile recharging stations. The UGVs and UAVs were organized into separate UGV-UAV teams, with the environment optimally divided into segments for each team to conduct cyclic, persistent surveillance of task points. Within each segment, the UGV-UAV team visited task points optimally, utilizing the Dantzig-Fulkerson-Johnson (DFJ) linear programming algorithm. These studies are limited in using specific optimization algorithms for UGV and UAV routing, missing out on the potential benefits of combining different algorithms. The combina-

tion could enhance solution quality and computational efficiency by leveraging the strengths of each algorithm together [66].

Some works in the literature have also considered re-planning UAV-UGV paths owing to dynamic changes in the environment. Ni et al. [71] introduced a 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. [72] 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. [73] tackled a real-time navigation problem for urban firefighting with UAVs and UGVs in GNSS-denied areas. They used Monte-Carlo localization to accurately determine the robots' positions in a pre-mapped environment, a Lazy Theta* algorithm for global path planning, and 3D-LIDAR mapping to enable the local path planner to quickly adjust plans 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.

Apart from single-method optimization approaches, there are multi-agent and hybrid optimization algorithm frameworks studied for these type of Vehicle Routing Problems (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. [67] 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. [68] introduced MultiAGent Metaheuristic Architecture (MAGMA), an optimization framework that facilitated cooperation between 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. [78] 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. 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.

There are some learning based approaches utilized in other multi-agent optimization frameworks. In Lotfi and Acan [79], 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. [80]. Some research has been done in exploring the capabilities of hyperheuristics for solving various combinatorial optimization problems. Cowling et al. [81] 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. [82] 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. [83] 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. [84] 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. [85], 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 simulations current state. This dynamic selection enhances real-time computational efficiency.

There are also some research work that's been done in implementing RL as hyper-heuristic agent for heuristic selection. Garrido et al. [86] 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 heuris-

tics (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. [87] 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. [88] 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.

While the above methods demonstrate promising results in applying RL within hyper-heuristic frameworks, existing literature primarily uses simpler heuristics as low-level algorithms

and is limited to basic instances of Vehicle Routing Problems. This paper advances these efforts by proposing an 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. The proposed method is evaluated against various approaches and tested on diverse problem sizes and distributions, demonstrating its generalizability. To this end, we present the following novel contributions:

1. We model the collaborative heterogeneous UAV-UGV routing problem as a bi-level optimization problem and solved using an RL-Assisted A-Teams optimization framework.
2. 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.
3. This approach is applied to a heterogeneous UAV-UGV system.
4. We also further extend the capability of existing multi-agent A-Teams framework through the proposed approach.

3 METHODS

3.1 Problem statement

To address the collaborative UGV-UAV routing problem for surveillance applications, we define a set of task points, $\mathcal{M} = \{m_0, m_1, \dots, m_n\}$, located within a guided road network. This UGV-UAV system consists of a set of UAVs $\mathcal{A} = \{u_i^a, \forall i = 1, \dots, K\}$ and one UGV persistently

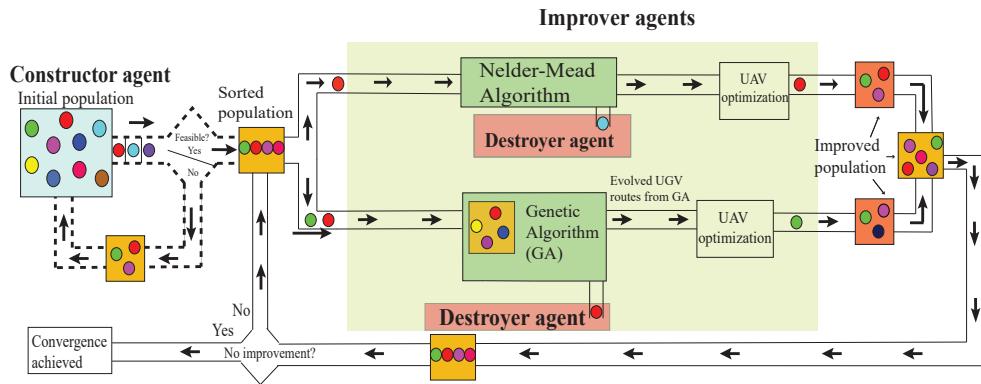


Figure 27. Description of the conventional A-Teams optimization framework used for UGV-UAV routing. This A-Teams inspired from the works of Talukdar et. al has Constructor, Improver, Destroyer and Predictor agents. Agents strategically choose UGV routes, which are fed into UAV optimization to get collaborative UGV-UAV route outputs.

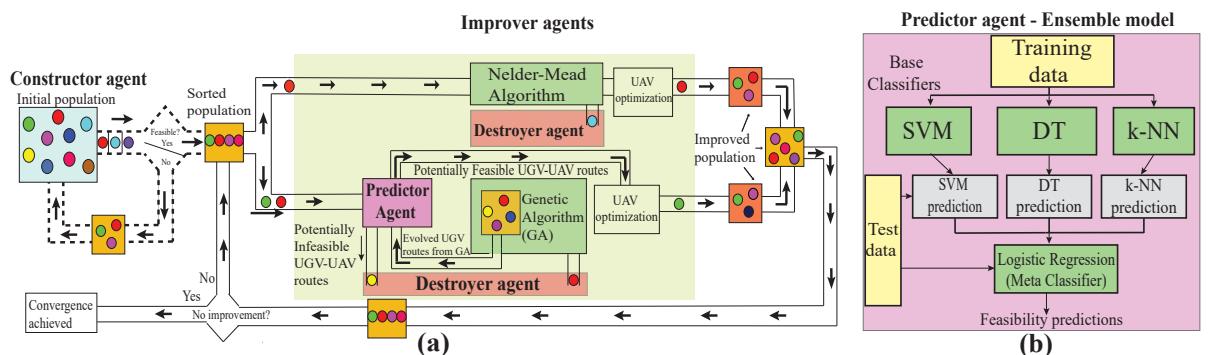


Figure 28. Description of the proposed A-Teams optimization framework used for UGV-UAV routing. a) This proposed A-Teams has Constructor, Improver, Destroyer and Predictor agents. Agents strategically choose UGV routes, which are fed into UAV optimization to get collaborative UGV-UAV route outputs. b) Training of Ensemble model in Predictor Agent

visiting \mathcal{M} . The UGV and UAV start from the depot D . As a realistic demonstrative application, these points are considered for persistent surveillance by both UAV and UGV. The UAV can either get recharged on UGV or at D . The vehicle system is of a heterogeneous nature where the UAV travels with higher velocity v_a but is limited in its energy capacity E_a . The approach taken in this study to extend the range of UAV is to pair it with a slower-moving UGV with velocity v_g , which has a larger energy capacity E_g to support UAV recharging and power its components. The operational range of the UAV with full energy E_a is visualized as blue circles in Figure 26 a), demonstrating that a single UAV cannot adequately cover all task points, thus substantiating the necessity for strategic UGV routing to ensure comprehensive UAV task coverage and efficient rendezvous. This interdependence between the UAV and UGV routes requires a bi-level optimization approach: the UGV route is optimized at the outer level, followed by the UAV route at the inner level, implementing a "UGV first, UAV second" strategy.

The UGV route optimization is formulated as a set of free parameters, denoting the rendezvous locations that the UAV makes with the UGV. The complete parameter set for a candidate UGV route, X_s , is made up of two subsets: primary rendezvous points $X_p = \{x_1, x_2, \dots, x_j\}$, which facilitate UAV recharging and directing towards different task sub-regions, and mid-rendezvous points $X_m = \{x_1, x_2, \dots, x_k\}$, 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, as described by Mondal et al. [74]. This approach effectively addresses the NP-Hard nature of the MSC problem, allowing for mul-

tuple X_p sets depending on the scale and complexity of the scenario. Equations Equation 3.1–Equation 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 $S = |X_s| = j + k$. For each candidate UGV route, the inner-level UAV optimization is solved for near-optimal UAV routes. This UAV optimization is formulated as an Energy-Constrained Vehicle Routing Problem (E-VRP), subject to both UGV route constraints and UAV fuel limitations. The UAV route solution is then fed back to the outer level for UGV optimization, providing feedback on the feasibility of the UAV route for the given UGV path. Feasibility criteria require that the UAV visit all task points at least once, ensuring mission criteria satisfaction.

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

s.t.,

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

where

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

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 $a^d = t - t_{last}^d$ 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* as:

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

Here $t_0^d, t_1^d, \dots, t_l^d$ 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 *score metric* is quadratic in time, thus punishing nodes with longer visit intervals harshly. Hence, the **objective** of this persistent surveillance problem is to minimize this *score metric* so that all the nodes are visited frequently with minimal intervals between successive visits.

Figure 26 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 $\mathcal{R} = \{r_0, r_1, \dots, r_n\}$. 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 research examines four distinct methods, each focusing on different approaches for optimizing UGV routes while keeping the UAV route fixed as an Energy-Constrained Vehicle Routing Problem (E-VRP). One of these is the proposed method introduced in this work. Since the UAV route depends on the UGV route, improving the efficiency of UGV route optimization reduces the burden on UAV optimization, decreasing overall computational demand and making the entire process more efficient. The following describes the four methods, each featuring a unique approach to UGV route optimization.

3.2 Method 1: GA at outer-level and Constraint Programming (CP) at inner-level

Genetic algorithms (GAs) are meta-heuristic techniques inspired by natural selection, 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 depicting candidate UGV routes. For each UGV route sent, the inner-level block optimizes the UAV route using Google's OR-Tools solver [34]. OR-Tools uses local search heuristics and solves using the CP approach. Some of the constraints, such as time constraints on the UAV, 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 mission points during 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 [77]. More details about this can be found in our previous work [51].

3.3 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. The agents in the framework possess distinct functionalities, solve 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, bad, 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.

Figure 27 shows the optimization using A-Teams at outer-level UGV routing. For solving this collaborative UGV-UAV routing problem, our previous work [75] has implemented this framework for similar kind of scenarios 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 $\mathcal{N} = 40$. Those UGV route parameters are sent to the inner-level block and UAV optimization is solved using OR-Tools. Based upon sending the feasibility of the obtained UGV-UAV route as feedback, the Improver Agents improve the UGV route until convergence is obtained, therefore giving an optimal collaborative route solution. Here, Nelder-Mead and GA are used as Improver agents. For each tuned UGV

route parameter given, the UAV optimization is carried in the inner-level via Constrained Programming method using OR-Tools solver.

3.4 A-Teams with Predictor Agent at outer-level and CP at inner-level

3.4.1 Solving outer-level UGV routing using Asynchronous Teams (A-Teams) framework with Predictor Agent

Figure 28 a) represents the proposed framework to perform UGV-UAV optimization. A-Teams is a multi-agent framework that uses a team of algorithms to optimize a given problem. The agents in the framework possess distinct functionalities and solve a problem through cooperation and achieve better solutions than their individual counterparts. There are four agents: **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 the existing A-Teams framework can be found in [75].

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 $\mathcal{N} = 40$. These routes are sent to the inner-level block for UAV optimization 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 28 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 visited at least once. The Ensemble model, tested with UGV routes generated by a Genetic Algorithm, filters out infeasible routes before UAV optimization, thereby enhancing process efficiency.

3.5 Reinforcement Learning framework for High-level Decision Making

This section outlines the Deep Reinforcement Learning (DRL) implemented on existing 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. Hence, in this work, we propose a DRL-based approach that operates as a high-level decision-making system that dynamically learns an optimal policy for selecting algorithms to execute at each stage of the optimization process. 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.

To implement the DRL, this can be modeled 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 $\langle State, Actions, Reward, Policy \rangle$. The flowchart representation of MDP is defined in Figure 29. 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: **Rendezvous points**, indicating the optimal UAV-UGV rendezvous locations for the current iteration's best solution obtained; **Local best**, representing

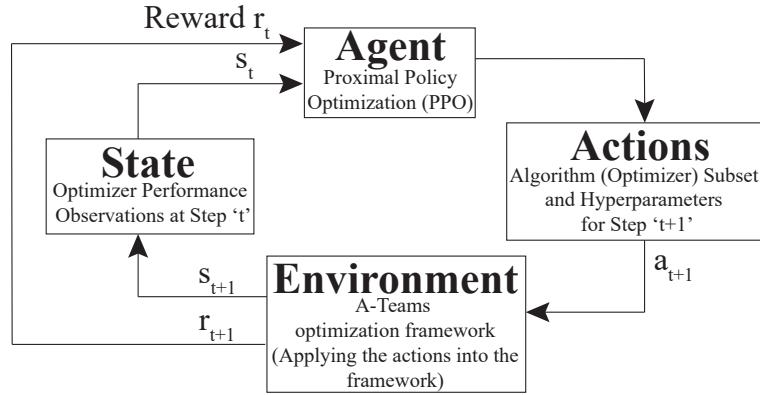


Figure 29. Markov Decision Process in this study

the current best solution from the local optimizer; **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; **Global best**, representing the current best solution from the global optimizer; **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; **Current best**, representing the best solution of the current state; **Action step**, representing the action chosen for this time step to generate the current observations; **Population size**, representing the number of UAV-UGV optimizations performed; **Local optimizer size**, representing the number of UAV-UGV optimizations performed by local optimizer; **Global optimizer size**, representing the number of UAV-UGV optimizations performed by global optimizer; **Predictor accuracy**, representing the accuracy of the ML model used for predictions; **Global best without Predictor agent**, representing the best solution of current state from global optimizer without using Ensemble

Figure 30. Action space for DRL based approach

learning model. This helps to gauge the quality of the Predictor Agent used in the optimization; **Predictor agent flag**, representing the binary variable to see if the Predictor Agent is used in the current state or not.

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 thus constituting to a total set of 8 discrete actions. The graphical illustration of the action space is depicted in Figure 30.

3) Reward: 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 computationally expensive, PPO is utilized because of its nature of simplicity and sample efficient [89].

3.5.1 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:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} (r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right] \quad (3.5)$$

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

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \quad (3.6)$$

In this equation:

- $\pi_\theta(a_t|s_t)$ is the probability of action a_t under the current policy θ 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.

The clipping function, $\text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)$, restricts the value of $r_t(\theta)$ within the interval $[1 - \epsilon, 1 + \epsilon]$, where ϵ 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 .

3.5.2 Advantage Estimation

The advantage function \hat{A}_t is estimated through the temporal difference (TD) error:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \dots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \quad (3.7)$$

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

3.5.3 Value Function Loss

PPO uses an additional loss term for the value function to improve the critics accuracy.

The value function loss is typically defined as:

$$L^{\text{VF}}(\theta) = \mathbb{E}_t \left[\left(V_\theta(s_t) - V_t^{\text{target}} \right)^2 \right] \quad (3.8)$$

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

3.5.4 Total Objective Function

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

$$L(\theta) = \mathbb{E}_t \left[L^{\text{CLIP}}(\theta) - c_1 L^{\text{VF}}(\theta) + c_2 S[\pi_\theta](s_t) \right] \quad (3.9)$$

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.

3.5.5 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 *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 5

Algorithm 5 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
         $\hat{A}_1, \dots, \hat{A}_T$ 
        Optimize surrogate L with respect to  $\theta$ , with K epochs and minibatch size  $M \leq NT$  Update
         $\theta_{\text{old}} \leftarrow \theta$ 

```

In this study, the limit for *T* was kept to be 4 to run the entire optimization cycle up to 4 iterations if the convergence for optimal solution is not reached within the limit, number of epochs *K* 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 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.

3.5.6 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 fuel 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 $V = \{0, 1, 2, \dots, m, D\}$ 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 UAV could utilize to get recharged. Let c_{ij} be the non-negative arc cost between a particular i and j . Let t_{ij} be the time travel 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. We formulate the VRP problem with fuel constraints, time windows, and dropped visits. The mathematical formulation for EVRP is presented here, but more details may be found in [51].

Objective:

$$\min \sum_{i \in V} \sum_{j \in V} t_{ij} x_{ij} \quad (3.10)$$

Major constraints:

$$f_j \leq f_i - (P^a t_{ij} x_{ij}) + L_1(1 - x_{ij}), \forall i \in V, j \in V \setminus D \quad (3.11)$$

$$f_j = Q, \quad \forall j \in D \quad (3.12)$$

$$0 \leq f_j \leq Q, \quad \forall j \in V \quad (3.13)$$

$$t_j \geq t_i + ((t_{ij} x_{ij})) - L_2(1 - x_{ij}), \forall i \in V, j \in V \quad (3.14)$$

$$t_j^l \leq t_j \leq t_j^u, \quad \forall j \in V \quad (3.15)$$

$$x_{ij} = 1 \rightarrow \sum_{i \in V \setminus D} x_{ji} = 1, \forall j \in D, \forall i \in V \setminus D \quad (3.16)$$

The objective is Eq. Equation 3.10 is to minimize the time to complete UAV routing. The constraint in Eq. Equation 3.11 is the Miller-Tucker-Zemlin (MTZ) formulation [18] for sub-tour elimination which enables that none of the UAVs are fully drained out while eliminating loops. P^a in this equation represents the UAV power consumption curve, which will be provided in Section 4. Constraint Eq. Equation 3.12 states that if the vertex is a recharging UGV stop, UGV must refuel the UAV to its full capacity Q . Constraint Eq. Equation 3.13 is the condition that the UAV's fuel at any vertex in V should be between 0 and maximum fuel capacity. Constraint Eq. Equation 3.14 denotes that the cumulative arrival time at j^{th} node is equal to the sum of cumulative time at the node i , t_i and the travel time between nodes i and j , $t_{ij} x_{ij}$. Constraint Eq. Equation 3.15 is the time window constraint that tells the vehicle to visit a certain vertex in the specified time window for that node. The constraint

in Eq. Equation 3.16 ensures that if any UAV comes to the refuel vertex to recharge, an arc must exist between that refuel 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. [17] conducted research that tested the 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. More details about this can be found in [17].

4 RESULTS

We used Python 3 for all the computations: the OpenAI’s Stable Baselines 3 Reinforcement Learning (RL) package is used for RL agent, the Ensemble of different classification algorithms (kNN, SVM, Decision Trees) used in the Predictor agent is from the Scikit-learn package, a custom-written Genetic Algorithm, and Nelder-Mead from the Scipy package for performing UGV free parameter optimization; and OR-Tools 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 a single UAV and UGV, with the UAV having a battery capacity of 4000 mAh (total energy of $E_a = 287.7\text{kJ}$) and the UGV having an energy capacity of $E_g = 25.01\text{MJ}$. The UAV and UGV are set to travel at velocities of $v_a = 10\text{m/s}$ and $v_g = 4.5\text{m/s}$, 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 $P_g = 464.8v_g + 356.3$ (referred from [76]). Based upon the scale of scenarios considered (14×14 km), solving the MSC coverage problem results in a candidate UGV route X_s with two primary rendezvous locations ($j = 2$) and three mid UGV-UAV rendezvous locations ($k = 3$), adding up to a total of 5 parameters ($\mathcal{S}=5$) for the UGV route.

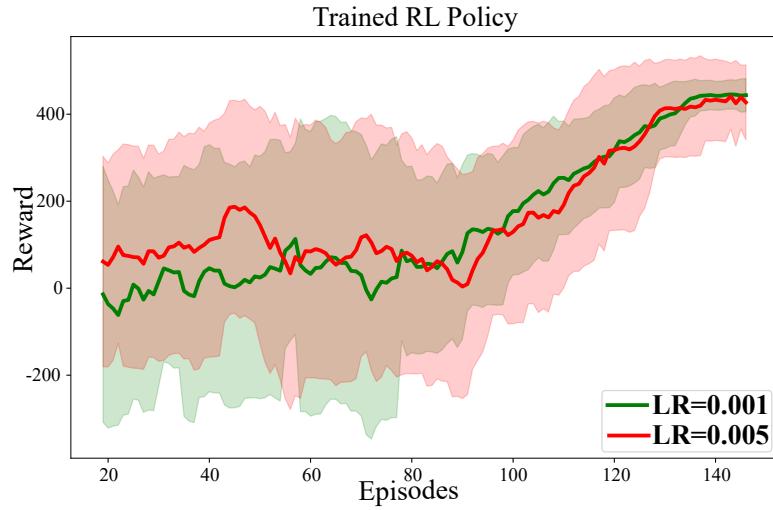


Figure 31. RL Training with different Learning Rates

The training for Reinforcement Learning policy is performed for several scenarios sent in batches. The scenario nature 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 32. Keeping the pattern similar, the RL model is trained with around 200 unique 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 the Figure 31. 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 the training for different learning rates, the RL model with $LR = 0.001$ has been chosen for testing. The training time with $LR = 0.001$ came out to be around 30 hours due to the inherent complexity in the learning mechanism of the RL policy. In terms of persistent surveillance, the end time of the mission T is kept to be 250 minutes.

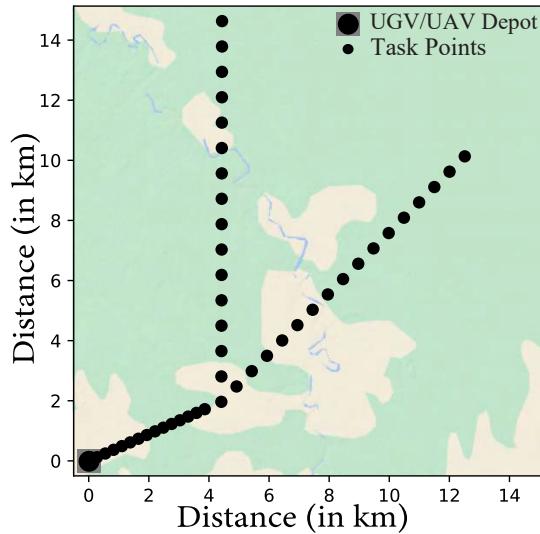


Figure 32. An instance of training scenario

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 33. This study is made with considering single UAV-UGV and multi UAV-UGV instances and the results are compared. Another study assess the ability of the proposed method to handle dynamic changes in the scenario. A case study scenario is being 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.

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, 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-

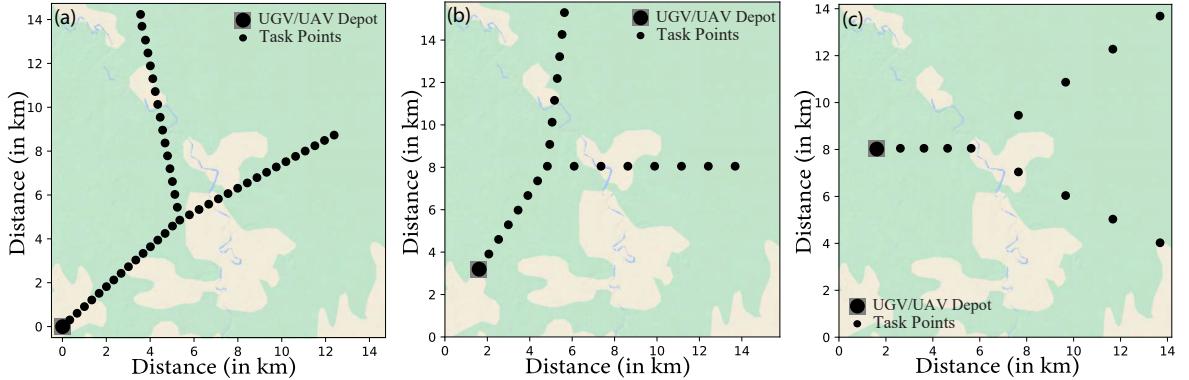


Figure 33. Example of different task point distribution scenario types. a) High Dense scenario
b) Moderate Dense scenario c) Sparse Dense scenario.

TABLE X

Comparison across different optimization methods for high-dense task point distribution with 1 UAV-1 UGV routing.

Scenario #	RL assisted A-Teams variant		A-Teams variant		Original A-Teams		Genetic Algorithm	
	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational 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	223.5	18	223.5	19	223.5	46	280.97

level. At the inner-level, the UAV optimization is performed by solving the E-VRP for all these methods.

TABLE XI

Comparison across different optimization methods for moderate-dense task point distribution with 1 UAV-1 UGV routing.

Scenario #	RL assisted A-Teams variant		A-Teams variant		Original A-Teams		Genetic Algorithm	
	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational 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

TABLE XII

Comparison across different optimization methods for sparse-dense task point distribution with 1 UAV-1 UGV routing.

Scenario #	RL assisted A-Teams variant		A-Teams variant		Original A-Teams		Genetic Algorithm	
	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational 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

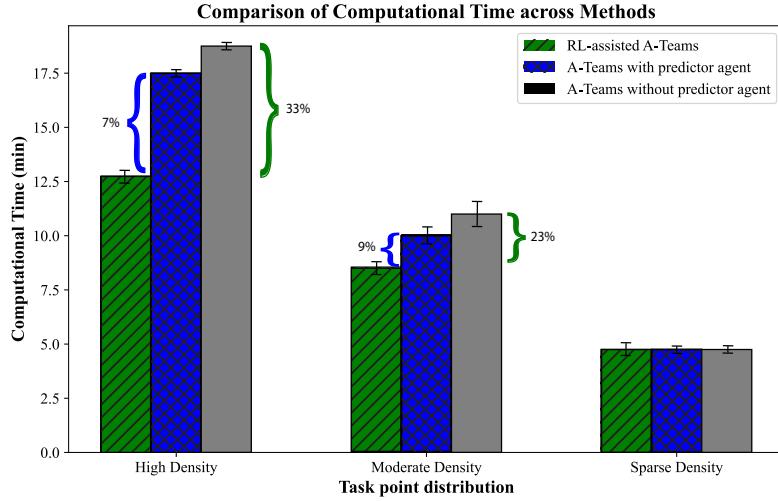


Figure 34. Computational time comparison across different methods: For 1 UAV - 1 UGV study

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

Table X through Table XII represents the computational analysis for various scenarios of 1 UAV - 1 UGV routing across different methods. 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 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 35. Also from Figure 34, 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.

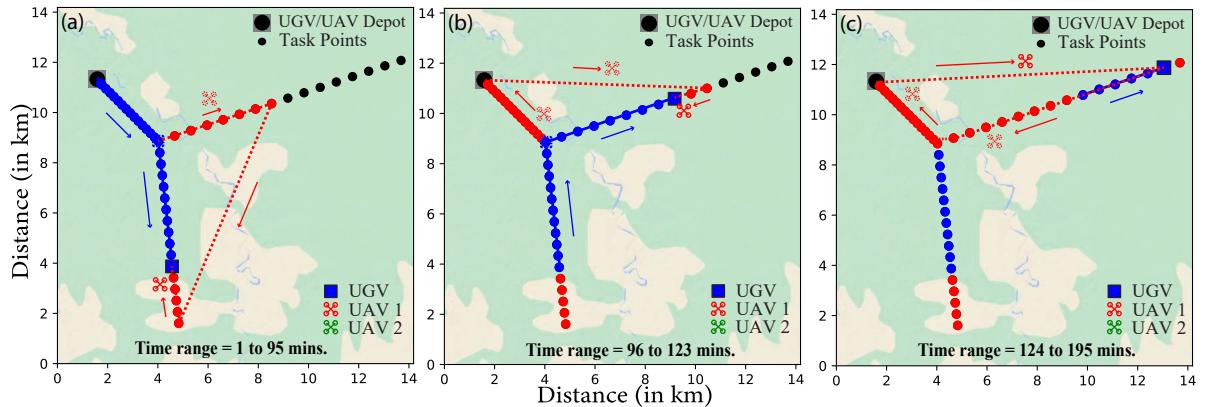


Figure 35. Route sequence of Scenario 1 of High Dense distribution with 1 UAV - 1 UGV routing

TABLE XIII

Comparison across different optimization methods for high-dense task point distribution with 2 UAV-1 UGV routing.

Scenario #	RL assisted A-Teams variant		A-Teams variant		Original A-Teams		Genetic Algorithm	
	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational 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

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 from the Table XIII through Table XV.

From the results 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 results between 1 UAV-UGV system and 2 UAV-UGV system, it can be seen that

TABLE XIV

Comparison across different optimization methods for moderate-dense task point distribution with 2 UAV-1 UGV routing.

Scenario #	RL assisted A-Teams variant		A-Teams variant		Original A-Teams		Genetic Algorithm	
	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational 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 XV

Comparison across different optimization methods for sparse-dense task point distribution with 2 UAV-1 UGV routing.

Scenario #	RL assisted A-Teams variant		A-Teams variant		Original A-Teams		Genetic Algorithm	
	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational time (min.)	Optimal Objective	Computational 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

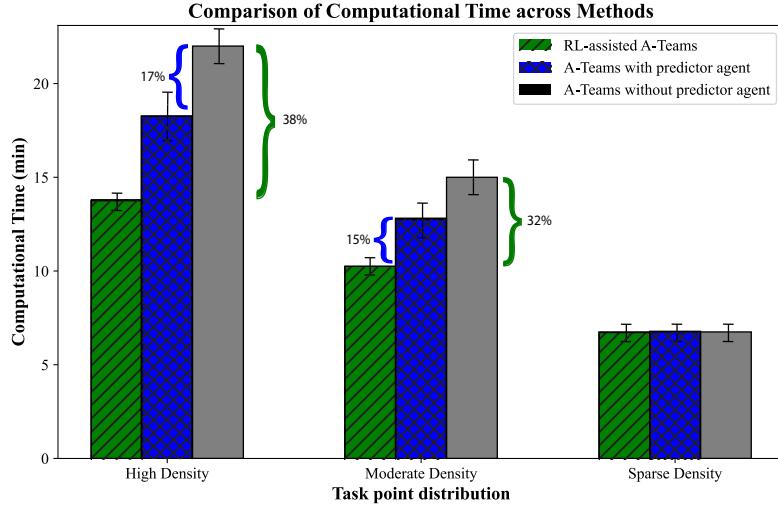


Figure 36. Computational time comparison across different methods: For 2 UAV - 1 UGV study

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 37. Also from Figure 36, 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 generalized to multi-UAV UGV systems.

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

From the above results, it is shown that the proposed Reinforcement Learning method solves the cooperative UAV-UGV routing problem with more computational efficiency as opposed to other methods. This opens up the potential for applying this method in a realistic environment.

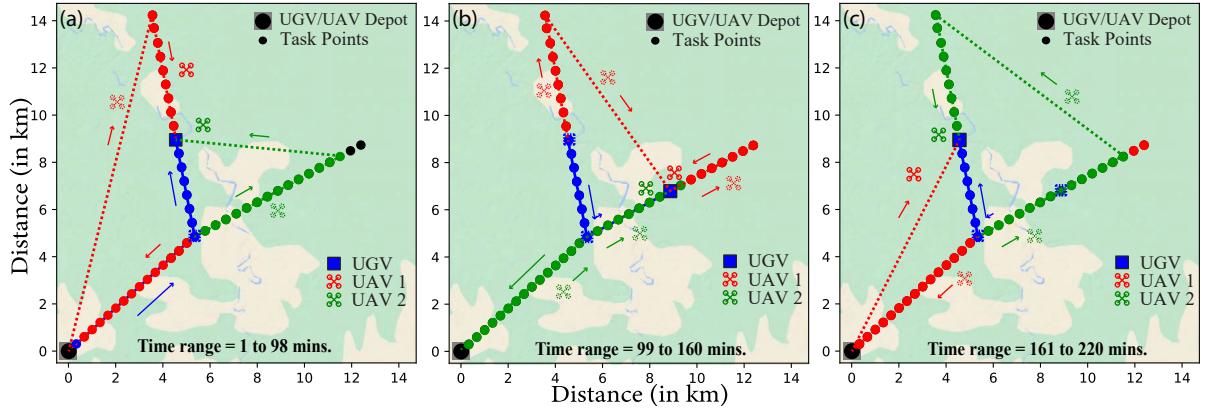


Figure 37. Route sequence of Scenario 1 of High Dense distribution with 2 UAV - 1 UGV routing

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 for infrastructure monitoring. Some of the works in the literature by Chan et al. [?] and Koch et al. [?] highlight the advantages of drone-based inspection over conventional methods, including cost reduction, faster inspection, and lesser traffic inconvenience to travelers. Typically, in these applications, drones are used for visual inspections by taking photos of the intersection point between the bridge deck and the columns. Notably, Omar et al. [?] investigate the use of drones equipped with infrared thermography for detecting subsurface delamination in concrete bridge decks, offering a non-contact 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 the drones are also capable of performing 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 a fully autonomous bridge inspection with minimal human intervention. This approach enhances efficiency, reduces inspection time, and improves safety throughout the process.

For this case study, a bridge section near Chicago, Illinois, US has been selected. This section spans approximately 8 kilometers and comprises three interconnected bridges: Long Run Creek, Des Plaines River Valley, and I-355 bridges. Fig. represents the bridge section under study. Initially, the bridge data is acquired from the ArcGIS map, providing essential information such as bridge details, latitude, and longitude. This data is then converted into Cartesian (x, y) coordinates, with the bottom-most point as the origin for reference. Fig. Figure 38 gives the graphical description of the Case Study scenario. Fig. a) represents the bridge locations obtained from the ArcGIS map. The blue points on the Fig. a) represents the bridges situated at those respective locations. Fig. b) represents the photo of the actual bridge. Fig. c) represents the converted plot of bridge locations into cartesian coordinates. The blue points are the points where UGV traverse to perform over-the-bridge inspection, and the red points are the points adjacent to the bridge deck where UAV flies to visit. These points are

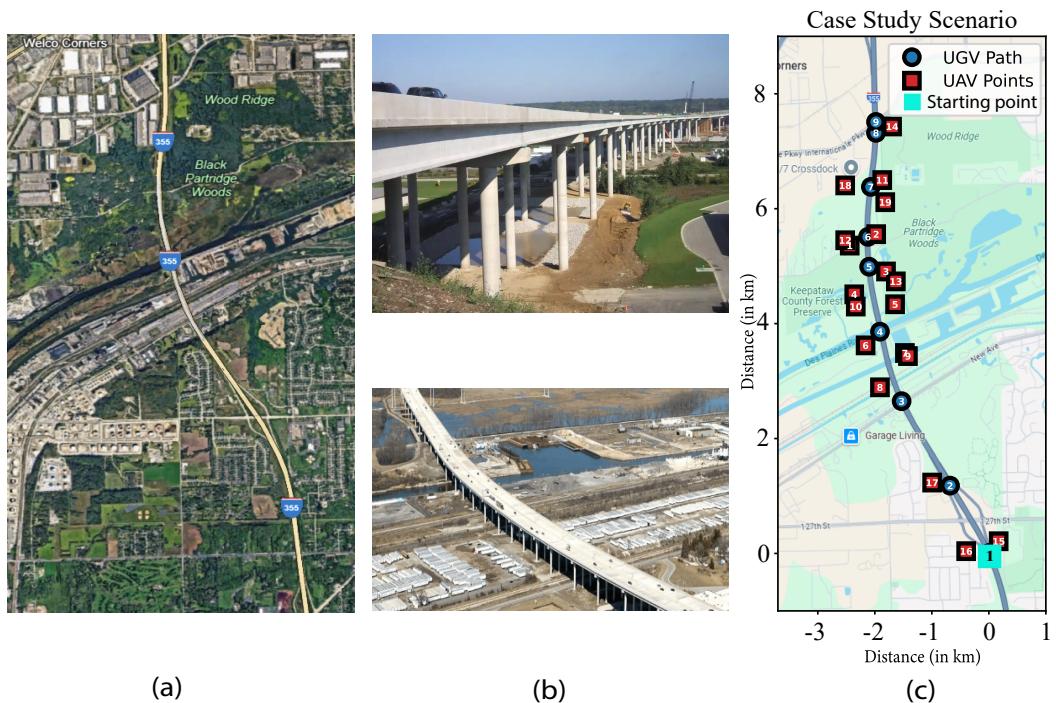


Figure 38. Case Study scenario considered for Bridge inspection. a) Bridge locations obtained from ArcGIS map b) Aerial view of the Bridge c) Converted coordinates for performing routing as per cartesian coordinates.

simulated to be the intersection points of the bridge deck and the bridge column to perform visual inspection. In order to form a road network, some additional points are added in the middle to represent the UGV points.

To set up a realistic simulation, the specifications of the UAV and UGV are as follows: UAV speed is kept to be 10 m/s based upon the work [?] and the UGV speed is kept to be 2.5 m/s to simulate the GPR traversal speed as per the reference here [?]. And typically, the visual inspection or the infrared inspection will take between 30 to 60 seconds at a particular point, and hence the inspection duration is considered to be 60 seconds in this study.

It is shown that the proposed RL-Assisted A-Teams method is better than the existing approaches, this method is used to perform the UAV-UGV routing. Persistent surveillance is performed for 150 minutes by the vehicle system. The UAVs are assumed to have both RGB camera and IR camera, hence they can do both thermal and visual inspection of the bridge. This enhances the generalizability of utilizing it for autonomous inspection.

To show the impact of using the autonomous vehicle team, the team configuration is varied across different UAVs. That is, a comparison analysis is performed between 1 UAV - 1 UGV system and 2 UAV - 1 UGV system where the respective system was implemented for performing the persistent surveillance for 150 minutes. The Age period of the task points to be visited is considered as the metric for the comparison. Fig. Figure 39 represents the age period heatmap of the points visited by the respective team. From the results, it can be seen that the 2 UAV - 1 UGV system performs better as it has lesser age period across the nodes compared to the 1 UAV - 1 UGV system. One of our previous research work shows the appropriate team configuration

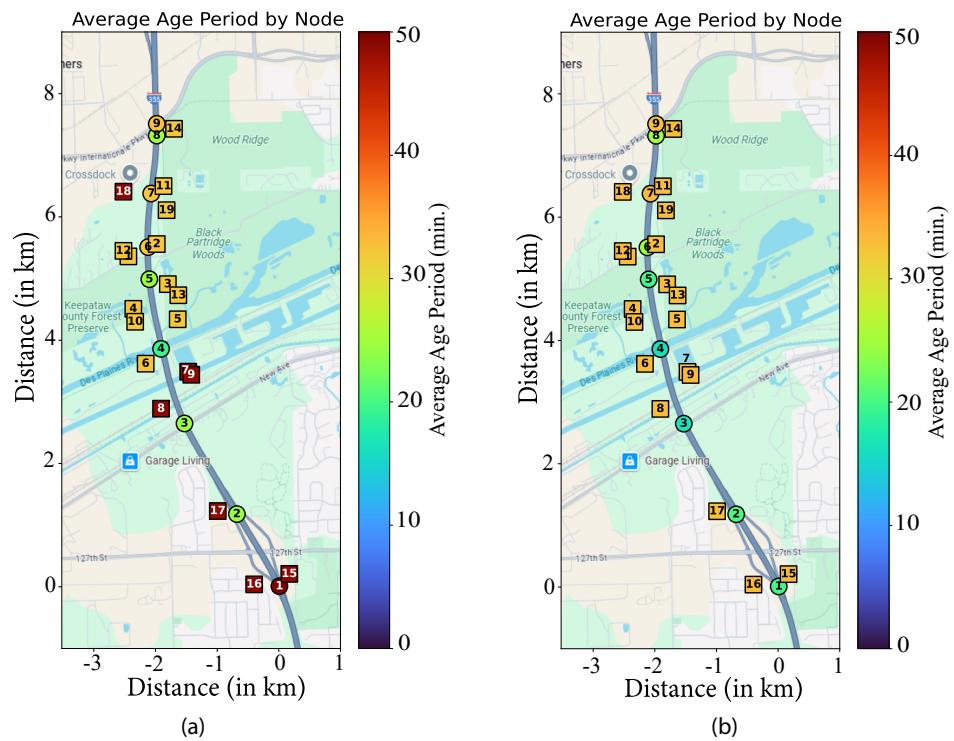


Figure 39. Average age period heatmap. a) For 1 UAV - 1 UGV system. b) For 2 UAV - 1 UGV system.

needed to have an overall better routing. The work shows that with a configuration ratio of number of vehicles being 2:1 for UAV:UGV system, the system performs better than others. Likewise, our configuration of 2 UAV - 1 UGV system is 2:1 and hence it performs better inspection than 1 UAV - 1 UGV system.

To test the UGV-UAV routing across different starting points to see the effect of routing, we have varied the starting to two different places: One case is starting the vehicle system from the bottom point, and the other case is starting them from the middle of the bridge. Figure Figure 40 a) and b) shows the different starting points. The routing comparison is done between these two cases for 1 UAV-UGV and 2 UAV-1UAV systems. Table Table XVI shows the average value of the average age period of across all the nodes for the respective vehicle system and different starting point. From the table it can be seen that, while the multi UAV-UGV system reduces the age period overall with more than 7 minutes reduction on average, the cases with bottom starting point performs better for both the vehicle system than starting from the middle.

To further investigate the potential of faster optimization capability of the proposed method, dynamic planning was considered for performing persistent surveillance, particularly when new mission points emerge randomly during the routing process. This dynamic planning was considered for 2 UAV - 1 UGV system. These new inspection points appearing randomly is assumed to be located outside the road network as those points are assumed to appear for the unexpected visual inspection of bridge columns by the UAVs. A critical assumption is that the UAV will come to know about the dynamic change information during their rendezvous with the

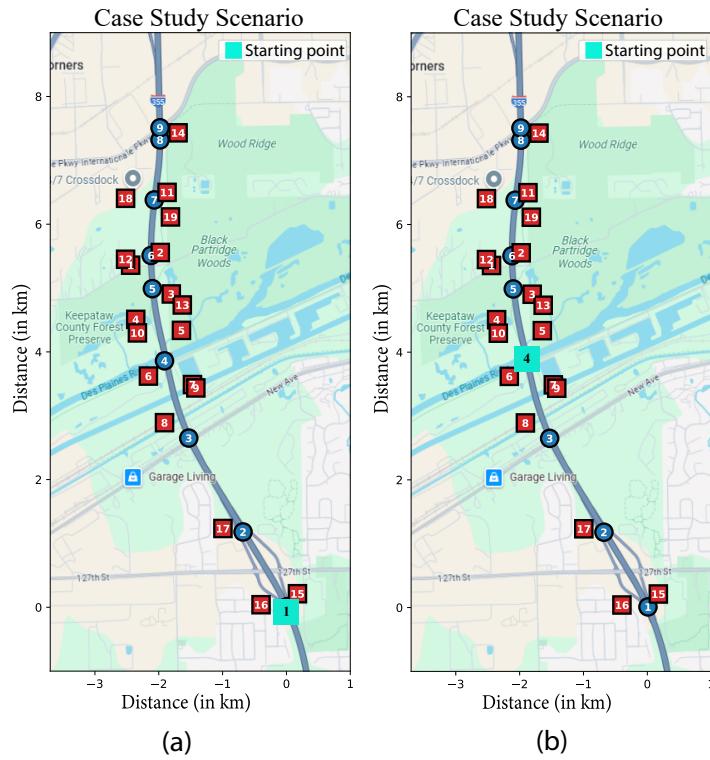


Figure 40. Case Study scenario with different starting points. The Green box in (a) and (b) represents the location from where the UAV-UGV inspection starts.

TABLE XVI

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

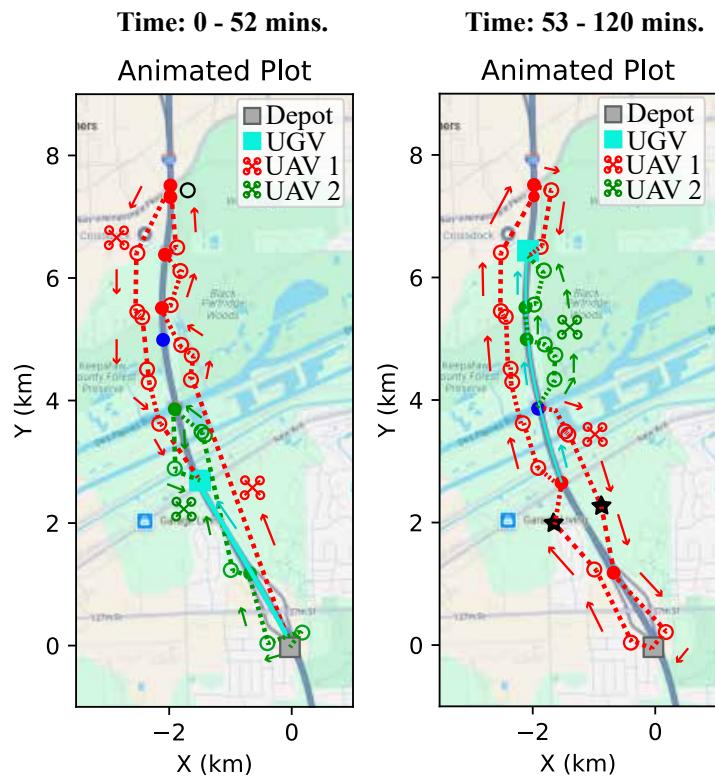


Figure 41. Case Study scenario routing sequence. Black stars on the right plot represent the dynamic appearance of inspection points to be visited.

UGV as UGV is the vehicle that is assumed to communicate such changes when it has made contact with the UAV. Fig. Figure 41 represents the route plan with dynamic changes. The star represents the dynamic points appeared during the timestamp between (enter the appropriate timestamp here), which is fed to the UAVs during their rendezvous with the UGV. And during the recharging time duration, the proposed optimization method re-plans the route and send the updated route to the UAVs before they takeoff after recharging. The re-optimization time was found to be 3 minutes on average, which is significantly faster than the recharging time of 15 minutes, showcasing the practical applicability of re-planning with dynamic changes. Figure Figure 42 represent the average age period of each node in the scenario for 2 UAV - 1 UGV system. The top plot represent the scenario routing without dynamic changes and the bottom plot represents the routing with dynamic changes. The influence of adding dynamic points increases the overall average age periods of each node because of the additional visits the UAV-UGV system needs to undergo.

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 pol-

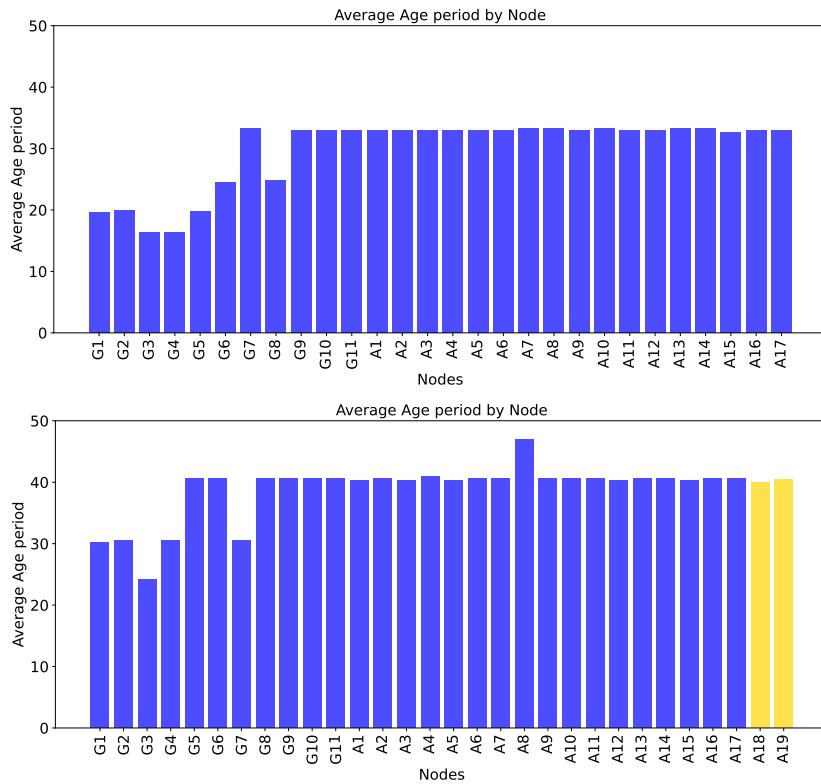


Figure 42. Average age period comparison across all the nodes. Top plot represent the 2 UAV - 1 UGV routing without dynamic changes and bottom plot represents the 2 UAV - 1 UGV routing with dynamic changes. The yellow bars represent the additionally added points (Black stars from Figure Figure 41).

icy 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 encouraged. 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.

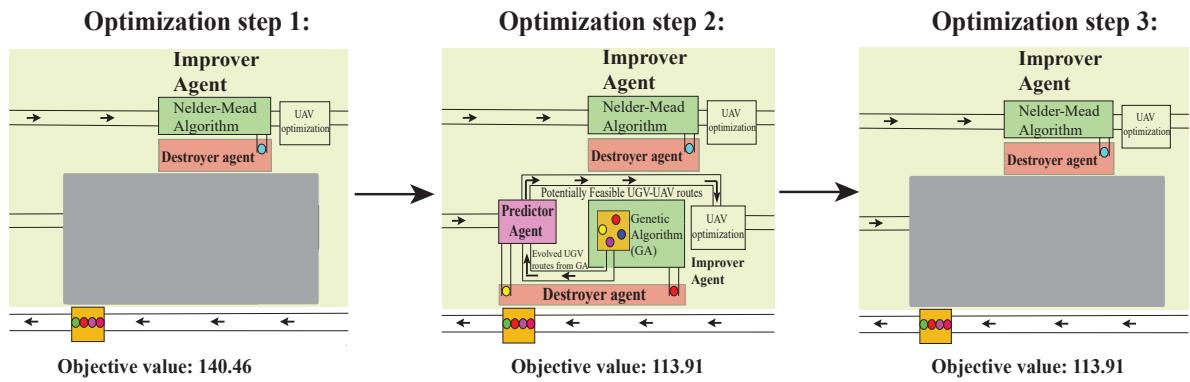


Figure 43. RL Policy-Driven Optimization Sequence for Agent Selection

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.

Within this framework, 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.

The key to obtaining 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 43, 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.

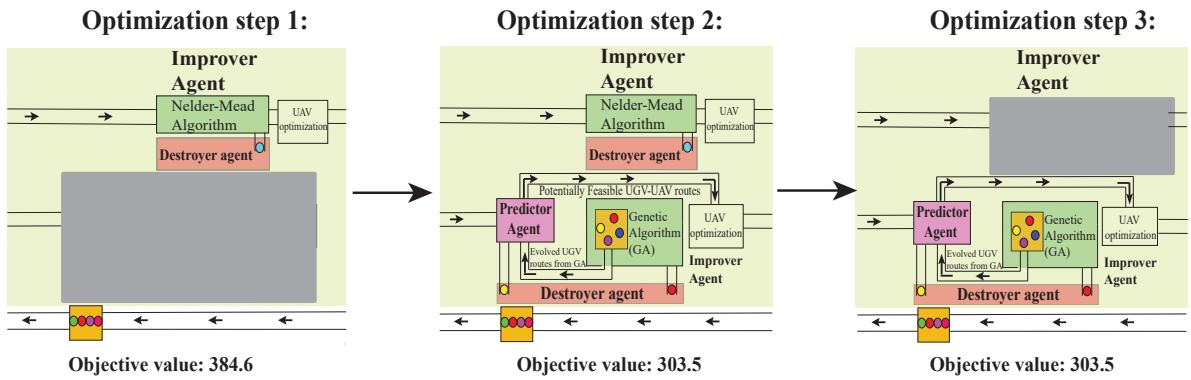


Figure 44. RL Policy-Driven Optimization Sequence for Agent Selection. Here Alternative reward mechanism is shaped to encourage choosing the Predictor agent as a part of optimization process.

If the optimization deems to prioritize 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 44 shows an optimization sequence 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 X 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 RL policy used in this work is a learning-based hyperheuristics, it only acts at the higher level. And relatively, the lower-level optimization process of performing the UAV optimization computationally expensive when compared to the typical training speed. Hence, the training process takes a lot of time despite using PPO method that is sample efficient. Also, with the involvement of several optimization agents in the framework, the reward shaping needs to be done critically in order to guide the policy's decision-making accordingly.

6 CONCLUSION

This study introduces a Reinforcement Learning-based hyperheuristic as a decision-maker for efficient optimization within the multi-agent A-Teams framework, benchmarking its performance against various A-Teams variants and a traditional Genetic Algorithm. The proposed method demonstrates the ability to produce near-optimal solutions more quickly than the com-

parison methods. Additionally, the framework's effectiveness was validated across different scenario instances.

Future work will focus on incorporating Large Language Models (LLMs) into the framework to automate action space selection, allowing the language model to choose actions based on text-based feedback provided by the framework.

CHAPTER 7

SOFTWARES USED

1 Optimization method

A significant portion of this work involves using Python Programming Language to develop the optimization method, with hardware testing also conducted through a Python-based API.

CHAPTER 8

CONCLUSION

Starting with the implementation of a simple optimization model and advancing to the development of a complex multi-agent optimization framework, this thesis explores the potential for creating an optimization method for collaborative UAV-UGV routing. The developed method showcases the potential of applying it on actual hardware system because of its ability to solve quickly.

Chapter 2 lays the premise for formulating the collaborative UAV-UGV routing problem as a Mixed Integer Linear Programming (MILP) bi-level optimization problem. This problem was approached using exact method algorithms. Exact methods, such as Branch and Bound, aim to find the true optimal solution; however, the complexity of these algorithms increases significantly with the size of the problem. Initial studies in this thesis revealed the substantial time and computational constraints involved in solving the problem using these exact methods.

Chapter 3 develops a method to solve the problem at a relatively faster pace. Since the exact solution methods are computationally expensive, it is not advisable to implement in practical scenarios dealing with such autonomous mobile robots. Hence, this thesis eventually directs towards implementing metaheuristic methods to solve the problem. This chapter of the thesis focuses on implementing a bi-level optimization approach, utilizing Genetic Algorithms and Bayesian Optimization for the outer-level UGV optimization and comparing these two methods, while employing local search heuristics for the inner-level UAV optimization. The

UGV route needs to be determined first as the UAV route involves recharging on the UGV. By suitably parameterizing the heuristics and optimizing the UGV route parameters with global optimization methods enables exploration of the space and provide good quality solutions.

Chapter 4 presents a method designed to efficiently solve the complex bi-level optimization problem by effectively leveraging the available computational resources. An optimization framework was developed for solving the UAV-UGV routing problem. The outer-level UGV optimization is dealt with developing a multi-agent optimization method that facilitates the capability to perform concurrent optimization with different set of agents that are complementary in nature. This multi-agent optimization framework is called A-Teams (Asynchronous Teams). The agents, containing optimization algorithms, are autonomous, modular and distributed in the framework. The study shows that the A-teams is able to produce good quality solutions with less computation time. It is able to do so by using specialized agents: agents to create solutions, agents to improve solutions globally and locally, and agents to destroy bad solutions.

Chapter 5 imparts the novelty in the existing A-Teams used, which improves the A-Teams optimization framework in terms of making it computationally efficient. By integrating a new component called the Predictor Agent, the framework incorporates machine learning capabilities into the route optimization process, thereby advancing its overall performance and effectiveness. The Predictor Agent implements a supervised Ensemble learning model with k-Nearest Neighbors, Support Vector Machines and Decision Trees. The developed framework is demonstrated to generate near-optimal solutions faster than the existing methods. This computational efficiency opens up the capabilities of this multi-agent optimization framework to be applied on

autonomous mobile robots operating in dynamically changing environments. Hence, it was validated in actual setting by conducting a hardware experiment on a case study scenario subjected to dynamic changes and the test was successful.

Chapter 6 explores strategies to achieve end-to-end autonomy in the optimization framework. It focuses on implementing a Reinforcement Learning (RL) policy as a high-level hyperheuristic that autonomously selects algorithms and their corresponding hyperparameters throughout the optimization process. This approach enhances computational efficiency and ensures the entire system operates independently. This RL-Assisted A-Teams method is put to test on a variety of problem instances and UAV-UGV team configuration and demonstrated the ability to produce near-optimal solutions more quickly than the previously developed methods. The proposed method demonstrates the significant impact of integrating Artificial Intelligence to perform efficient and autonomous optimization.

This approach can be further enhanced by integrating Large Language Models (LLMs) into the framework. LLMs can analyze performance metrics and historical data from the optimization algorithms to generate recommendations for optimal hyperparameter settings. By providing these intelligent suggestions, LLMs help reduce the decision-making burden on the RL policy, effectively distributing the tasks and improving the overall efficiency and autonomy of the optimization process.

Chapter 7 gives the information about the software used for the most of the thesis. Python is primarily utilized for accessing APIs, managing packages, and running custom scripts, while MATLAB and Python are employed to create plots and figures.

The project files can be found in my GitHub page: <https://github.com/Subram0212/Dissertation>

APPENDICES

Appendix A

COPYRIGHT PERMISSIONS

In this appendix, we present the copyright permissions for the articles, whose contents were used in this thesis.

Appendix A (Continued)

11/8/24, 11:22 PM

Rightslink® by Copyright Clearance Center



RightsLink



Coordinated Route Planning of Multiple Fuel-constrained Unmanned Aerial Systems with Recharging on an Unmanned Ground Vehicle for Mission Coverage

SPRINGER NATURE

Author: Subramanian Ramasamy et al
 Publication: Journal of Intelligent and Robotic Systems
 Publisher: Springer Nature
 Date: Sep 17, 2022

Copyright © 2022, The Author(s), under exclusive licence to Springer Nature B.V.

Order Completed

Thank you for your order.

This Agreement between Subramanian Ramasamy ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

License Number 5904600505375

[Printable Details](#)

License date Nov 09, 2024

Licensed Content

Licensed Content Publisher	Springer Nature
Licensed Content Publication	Journal of Intelligent and Robotic Systems
Licensed Content Title	Coordinated Route Planning of Multiple Fuel-constrained Unmanned Aerial Systems with Recharging on an Unmanned Ground Vehicle for Mission Coverage
Licensed Content Author	Subramanian Ramasamy et al
Licensed Content Date	Sep 17, 2022

Order Details

Type of Use	Thesis/Dissertation
Requestor Type	academic/university or research institute
Format	electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	1 - 29
Author of this content	Springer Nature
	yes

About Your Work

Title of new work	Developing Computationally Efficient Optimization Framework for Collaborative Routing of UAV-UGV System
Institution name	University of Illinois at Chicago
Expected presentation date	Nov 2024

Additional Data

The Requesting Person / Organization to Appear on the License	Subramanian Ramasamy
---	----------------------

Appendix B

COPYRIGHT PERMISSIONS

In this appendix, we present the copyright permissions for the articles, whose contents were used in this thesis.

Appendix B (Continued)



SR

Heterogenous vehicle routing: comparing parameter tuning using genetic algorithm and bayesian optimization

Conference Proceedings: 2022 International Conference on Unmanned Aircraft Systems (ICUAS)
 Author: Subramanian Ramasamy
 Publisher: IEEE
 Date: 21 June 2022

Copyright © 2022, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name] products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a license from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#)

[CLOSE WINDOW](#)

Appendix C

COPYRIGHT PERMISSIONS

In this appendix, we present the copyright permissions for the articles, whose contents were used in this thesis.

Appendix C (Continued)

 Requesting permission to reuse content from an IEEE publication

Solving Vehicle Routing Problem for Unmanned Heterogeneous Vehicle Systems using Asynchronous Multi-Agent Architecture (A-teams)
 Conference Proceedings: 2023 International Conference on Unmanned Aircraft Systems (ICUAS)
 Author: Subramanian Ramasamy
 Publisher: IEEE
 Date: 06 June 2023

Copyright © 2023, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a license from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#) [CLOSE WINDOW](#)

© 2024 Copyright - All Rights Reserved | [Copyright Clearance Center, Inc.](#) | [Privacy statement](#) | [Data Security and Privacy](#) | [For California Residents](#) | [Terms and Conditions](#)
 Comments? We would like to hear from you. E-mail us at customercare@copyright.com

Appendix D

COPYRIGHT PERMISSIONS

In this appendix, we present the copyright permissions for the articles, whose contents were used in this thesis.

Appendix D (Continued)


[Sign in/Register](#)


Computationally Efficient Multi-Agent Optimization Framework for Online Routing of UAV-UGV System



Conference Proceedings: 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)
 Author: Subramanian Ramasamy
 Publisher: IEEE
 Date: 28 August 2024

Copyright © 2024, IEEE

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

- 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
- 2) In the case of illustrations or tabular material, we require that the copyright line © [year of original publication] IEEE appear prominently with each reprinted figure and/or table.
- 3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

- 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
- 2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
- 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights_rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

[BACK](#)

[CLOSE WINDOW](#)

CITED LITERATURE

1. Maini, P., Sundar, K., Rathinam, S., and Sujit, P.: Cooperative planning for fuel-constrained aerial vehicles and ground-based refueling vehicles for large-scale coverage. *arXiv preprint arXiv:1805.04417*, 2018.
2. De Backer, B., Furnon, V., Shaw, P., Kilby, P., and Prosser, P.: Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics*, 6(4):501–523, 2000.
3. Khuller, S., Malekian, A., and Mestre, J.: To fill or not to fill: The gas station problem. *ACM Transactions on Algorithms (TALG)*, 7(3):1–16, 2011.
4. Kannon, T. E., Nurre, S. G., Lunday, B. J., and Hill, R. R.: The aircraft routing problem with refueling. *Optimization Letters*, 9(8):1609–1624, 2015.
5. Levy, D., Sundar, K., and Rathinam, S.: Heuristics for routing heterogeneous unmanned vehicles with fuel constraints. *Mathematical Problems in Engineering*, 2014, 2014.
6. Sundar, K. and Rathinam, S.: Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots. *IEEE Transactions on Automation Science and Engineering*, 11(1):287–294, 2013.
7. Song, B. D., Kim, J., and Morrison, J. R.: Rolling horizon path planning of an autonomous system of uavs for persistent cooperative service: Milp formulation and efficient heuristics. *Journal of Intelligent & Robotic Systems*, 84(1-4):241–258, 2016.
8. Younghoon, C., Youngjun, C., Briceno, S., and Mavris, D. N.: Energy-constrained multi-uav coverage path planning for an aerial imagery mission using column generation. *Journal of Intelligent & Robotic Systems*, 97(1):125–139, 2020.
9. Radzki, G., Golinska-Dawson, P., Bocewicz, G., and Banaszak, Z.: Modelling robust delivery scenarios for a fleet of unmanned aerial vehicles in disaster relief missions. *Journal of Intelligent & Robotic Systems*, 103(4):1–18, 2021.

10. Lee, A. C., Dahan, M., Weinert, A. J., and Amin, S.: Leveraging suas for infrastructure network exploration and failure isolation. *Journal of Intelligent & Robotic Systems*, 93(1-2):385–413, 2019.
11. Albert, A. and Imsland, L.: Combined optimal control and combinatorial optimization for searching and tracking using an unmanned aerial vehicle. *Journal of Intelligent & Robotic Systems*, 95(2):691–706, 2019.
12. Manyam, S. G., Casbeer, D. W., and Sundar, K.: Path planning for cooperative routing of air-ground vehicles. In *2016 American Control Conference (ACC)*, pages 4630–4635, 2016.
13. Petitprez, E., Georges, F., Raballand, N., and Bertrand, S.: Deployment optimization of a fleet of drones for routine inspection of networks of linear infrastructures. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 303–310, 2021.
14. Liu, Y., Liu, Z., Shi, J., Wu, G., and Chen, C.: Optimization of base location and patrol routes for unmanned aerial vehicles in border intelligence, surveillance, and reconnaissance. *Journal of Advanced Transportation*, 2019, 2019.
15. Bard, J. F., Jarrah, A. I., and Zan, J.: Validating vehicle routing zone construction using monte carlo simulation. *European Journal of Operational Research*, 206(1):73–85, 2010.
16. Dondo, R. and Cerdá, J.: A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European journal of operational research*, 176(3):1478–1507, 2007.
17. Ramasamy, S., Reddinger, J.-P. F., Dotterweich, J. M., Childers, M. A., and Bhounsule, P. A.: 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):1–18, 2022.
18. Miller, C. E., Tucker, A., and Zemlin, R. A.: Integer programming formulation of traveling salesman problems. *J. ACM*, 7:326–329, 1960.
19. Gurobi: Gurobi Optimization LLC. <https://www.gurobi.com/>, 2021. Online; accessed Sep 19, 2021.

20. Altshuler, Y., Pentland, A., and Bruckstein, A. M.: Optimal dynamic coverage infrastructure for large-scale fleets of reconnaissance UAVs. In Swarms and Network Intelligence in Search, pages 207–238. Springer, 2018.
21. Griffiths, S. R.: Remote terrain navigation for unmanned air vehicles. 2006.
22. Theile, M., Bayerlein, H., Nai, R., Gesbert, D., and Caccamo, M.: UAV coverage path planning under varying power constraints using deep reinforcement learning. arXiv preprint arXiv:2003.02609, 2020.
23. Grocholsky, B., Keller, J., Kumar, V., and Pappas, G.: Cooperative air and ground surveillance. IEEE Robotics & Automation Magazine, 13(3):16–25, 2006.
24. Ghamry, K. A., Kamel, M. A., and Zhang, Y.: Cooperative forest monitoring and fire detection using a team of uavs-ugvs. In 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pages 1206–1211. IEEE, 2016.
25. Wu, Y., Wu, S., and Hu, X.: Cooperative path planning of uavs & ugv for a persistent surveillance task in urban environments. IEEE Internet of Things Journal, 8(6):4906–4919, 2020.
26. Baldacci, R., Battarra, M., and Vigo, D.: Routing a heterogeneous fleet of vehicles. In The vehicle routing problem: latest advances and new challenges, pages 3–27. Springer, 2008.
27. Sundar, K., Venkatachalam, S., and Rathinam, S.: Formulations and algorithms for the multiple depot, fuel-constrained, multiple vehicle routing problem. In 2016 American Control Conference (ACC), pages 6489–6494. IEEE, 2016.
28. Ren, S., Chen, Y., Xiong, L., Chen, Z., and Chen, M.: Path planning for the marsupial double-uavs system in air-ground collaborative application. In 2018 37th Chinese Control Conference (CCC), pages 5420–5425. IEEE, 2018.
29. Mathew, N., Smith, S. L., and Waslander, S. L.: Multirobot rendezvous planning for recharging in persistent tasks. IEEE Transactions on Robotics, 31(1):128–142, 2015.
30. Ramasamy, S., Reddinger, J.-P. F., Dotterweich, J. M., Childers, M. A., and Bhounsule, P. A.: Cooperative route planning of multiple fuel-constrained unmanned aerial

- vehicles with recharging on an unmanned ground vehicle. In 2021 International Conference on Unmanned Aircraft Systems (ICUAS), pages 155–164. IEEE, 2021.
31. Huang, C., Yuan, B., Li, Y., and Yao, X.: Automatic parameter tuning using bayesian optimization method. In 2019 IEEE Congress on Evolutionary Computation (CEC), pages 2090–2097. IEEE, 2019.
 32. Henrio, J., Deligne, T., Nakashima, T., and Watanabe, T.: Route planning for multiple surveillance autonomous drones using a discrete firefly algorithm and a bayesian optimization method. Artificial Life and Robotics, 24(1):100–105, 2019.
 33. Pilat, M. L. and White, T.: Using genetic algorithms to optimize acs-tsp. In International workshop on ant algorithms, pages 282–287. Springer, 2002.
 34. Google: Google OR-tools. <https://developers.google.com/optimization>, 2021. Online; accessed Feb 2, 2021.
 35. Rossi, F., Van Beek, P., and Walsh, T.: Handbook of constraint programming. Elsevier, 2006.
 36. Shaw, P., Furnon, V., and De Backer, B.: A constraint programming toolkit for local search. In Optimization software class libraries, pages 219–261. Springer, 2003.
 37. Tatsch, C. A. A.: Route Planning for Long-Term Robotics Missions. West Virginia University, 2020.
 38. Voudouris, C. and Tsang, E. P.: Guided local search. In Handbook of metaheuristics, pages 185–218. Springer, 2003.
 39. Voudouris, C., Tsang, E. P., and Alshedy, A.: Guided local search. In Handbook of metaheuristics, pages 321–361. Springer, 2010.
 40. Wang, Q.: Using genetic algorithms to optimise model parameters. Environmental Modelling & Software, 12(1):27–34, 1997.
 41. McKay, M. D., Beckman, R. J., and Conover, W. J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics, 42(1):55–61, 2000.

42. Shukla, A., Pandey, H. M., and Mehrotra, D.: Comparative review of selection techniques in genetic algorithm. In 2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE), pages 515–519. IEEE, 2015.
43. Sokolov, A. and Whitley, D.: Unbiased tournament selection. In Proceedings of the 7th annual conference on Genetic and evolutionary computation, pages 1131–1138, 2005.
44. Song, Y. and Scaramuzza, D.: Policy search for model predictive control with application to agile drone flight. IEEE Transactions on Robotics, 2022.
45. Hu, D., Gan, V. J., Wang, T., and Ma, L.: Multi-agent robotic system (mars) for uav-ugv path planning and automatic sensory data collection in cluttered environments. Building and Environment, 221:109349, 2022.
46. Gao, W., Luo, J., Zhang, W., Yuan, W., and Liao, Z.: Commanding cooperative ugv-uav with nested vehicle routing for emergency resource delivery. IEEE Access, 8:215691–215704, 2020.
47. Lakas, A., Belkhouch, B., Benkraouda, O., Shuaib, A., and Alasmawi, H. J.: A framework for a cooperative uav-ugv system for path discovery and planning. In 2018 International Conference on Innovations in Information Technology (IIT), pages 42–46. IEEE, 2018.
48. Girma, A., Bahadori, N., Sarkar, M., Tadewos, T. G., Behnia, M. R., Mahmoud, M. N., Karimoddini, A., and Homaifar, A.: Iot-enabled autonomous system collaboration for disaster-area management. IEEE/CAA Journal of Automatica Sinica, 7(5):1249–1262, 2020.
49. Venkatachalam, S., Sundar, K., and Rathinam, S.: A two-stage approach for routing multiple unmanned aerial vehicles with stochastic fuel consumption. Sensors, 18(11):3756, 2018.
50. Mondal, M. S., Ramasamy, S., and Bhounsule, P.: A bilevel optimization framework for fuel-constrained uav-ugv cooperative routing: Planning and experimental validation. arXiv preprint arXiv:2303.02315, 2023.
51. Ramasamy, S., Mondal, M. S., Reddinger, J.-P. F., Dotterweich, J. M., Humann, J. D., Childers, M. A., and Bhounsule, P. A.: Heterogenous vehicle routing: compar-

- ing parameter tuning using genetic algorithm and bayesian optimization. In 2022 International Conference on Unmanned Aircraft Systems (ICUAS), pages 104–113. IEEE, 2022.
52. Boyd, S., Boyd, S. P., and Vandenberghe, L.: Convex optimization. Cambridge university press, 2004.
 53. Sachdev, S.: Explorations in asynchronous teams. Doctoral dissertation, Carnegie Mellon University, 1998.
 54. Talukdar, S., Ramesh, V., et al.: Cooperative methods for security planning. 1992.
 55. Jkederzejowicz, P. and Ratajczak-Ropel, E.: Reinforcement learning strategy for solving the resource-constrained project scheduling problem by a team of a-teams. In Asian Conference on Intelligent Information and Database Systems, pages 197–206. Springer, 2014.
 56. Kazemi, A., Fazel Zarandi, M., and Moattar Husseini, S.: A multi-agent system to solve the production–distribution planning problem for a supply chain: a genetic algorithm approach. The International Journal of Advanced Manufacturing Technology, 44(1):180–193, 2009.
 57. Jedrzejowicz, P. and Ratajczak-Ropel, E.: Experimental evaluation of a-teams solving resource availability cost problem. In Intelligent Decision Technologies 2019, pages 213–223. Springer, 2020.
 58. Rabak, C. S. and Sichman, J. S.: Using a-teams to optimize automatic insertion of electronic components. Advanced Engineering Informatics, 17(2):95–106, 2003.
 59. Barbucha, D., Czarnowski, I., Jkederzejowicz, P., Ratajczak-Ropel, E., and Wierzbowska, I.: Team of a-teams-a study of the cooperation between program agents solving difficult optimization problems. In Agent-based optimization, pages 123–141. Springer, 2013.
 60. Rachlin, J., Goodwin, R., Murthy, S., Akkiraju, R., Wu, F., Kumaran, S., and Das, R.: A-teams: An agent architecture for optimization and decision-support. In International Workshop on Agent Theories, Architectures, and Languages, pages 261–276. Springer, 1999.

61. Maini, P., Sundar, K., Singh, M., Rathinam, S., and Sujit, P.: Cooperative aerial-ground vehicle route planning with fuel constraints for coverage applications. *IEEE Transactions on Aerospace and Electronic Systems*, 55(6):3016–3028, 2019.
62. Tokekar, P., Vander Hook, J., Mulla, D., and Isler, V.: Sensor planning for a symbiotic uav and ugv system for precision agriculture. *IEEE transactions on robotics*, 32(6):1498–1511, 2016.
63. Zhang, M., Liang, H., and Zhou, P.: Cooperative route planning for fuel-constrained ugv-uav exploration. In *2022 IEEE International Conference on Unmanned Systems (ICUS)*, pages 1047–1052. IEEE, 2022.
64. Liu, Y., Luo, Z., Liu, Z., Shi, J., and Cheng, G.: Cooperative routing problem for ground vehicle and unmanned aerial vehicle: The application on intelligence, surveillance, and reconnaissance missions. *IEEE Access*, 7:63504–63518, 2019.
65. Seyedi, S., Yazıcıoğlu, Y., and Aksaray, D.: Persistent surveillance with energy-constrained uavs and mobile charging stations. *IFAC-PapersOnLine*, 52(20):193–198, 2019.
66. Blum, C., Puchinger, J., Raidl, G. R., and Roli, A.: Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.
67. Jedrzejowicz, P. and Ratajczak-Ropel, E.: 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.
68. Milano, M. and Roli, A.: Magma: a multiagent architecture for metaheuristics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(2):925–941, 2004.
69. Nigam, N., Bieniawski, S., Kroo, I., and Vian, J.: Control of multiple uavs for persistent surveillance: Algorithm description and hardware demonstration. In *AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference*, page 1852, 2009.
70. Karapetyan, N., Asghar, A. B., Bhaskar, A., Shi, G., Manocha, D., and Tokekar, P.: Ag-cvg: Coverage planning with a mobile recharging ugv and an energy-constrained uav. *arXiv preprint arXiv:2310.07621*, 2023.

71. Ni, J., Wang, X., Tang, M., Cao, W., Shi, P., and Yang, S. X.: An improved real-time path planning method based on dragonfly algorithm for heterogeneous multi-robot system. *IEEE Access*, 8:140558–140568, 2020.
72. Ma, N., Cao, Y., Wang, X., Wang, Z., and Sun, H.: 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.
73. Martínez-Rozas, S., Rey, R., Alejo, D., Acedo, D., Cobano, J. A., Rodríguez-Ramos, A., Campoy, P., Merino, L., and Caballero, F.: An aerial/ground robot team for autonomous firefighting in urban gnss-denied scenarios. 2022.
74. Mondal, M. S., Ramasamy, S., Humann, J. D., Reddinger, J.-P. F., Dotterweich, J. M., Childers, M. A., and Bhounsule, P. A.: Cooperative multi-agent planning framework for fuel constrained uav-ugv routing problem, 2023.
75. Ramasamy, S., Mondal, M. S., Reddinger, J.-P. F., Dotterweich, J. M., Humann, J. D., Childers, M. A., and Bhounsule, P. A.: 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.
76. Hurwitz, A. M., Dotterweich, J. M., and Rocks, T. A.: 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.
77. Mullen, K., Ardia, D., Gil, D. L., Windover, D., and Cline, J.: Deoptim: An r package for global optimization by differential evolution. *Journal of Statistical Software*, 40(6):1–26, 2011.
78. Ramasamy, S., Mondal, M. S., Humann, J. D., Dotterweich, J. M., Reddinger, J.-P. F., Childers, M. A., and Bhounsule, P. A.: 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.
79. Lotfi, N. and Acan, A.: 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.

80. Cowling, P., Kendall, G., and Soubeiga, E.: 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.
81. Cowling, P., Kendall, G., and Han, L.: 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.
82. Grobler, J., Engelbrecht, A. P., Kendall, G., and Yadavalli, V. S.: Heuristic space diversity control for improved meta-hyper-heuristic performance. Information Sciences, 300:49–62, 2015.
83. Sabar, N. R., Ayob, M., Kendall, G., and Qu, R.: A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. IEEE Transactions on Cybernetics, 45(2):306–315, 2015.
84. Lagoudakis, M. G., Littman, M. L., et al.: Algorithm selection using reinforcement learning. In ICML, pages 511–518, 2000.
85. Armstrong, W., Christen, P., McCreathe, E., and Rendell, A. P.: Dynamic algorithm selection using reinforcement learning. In 2006 international workshop on integrating ai and data mining, pages 18–25. IEEE, 2006.
86. Garrido-Castro, R., Turke, A., and van Woensel, T.: A flexible and adaptive hyper-heuristic approach for (dynamic) capacitated vehicle routing problems. European Journal of Operational Research, 222(1):111–122, 2012.
87. Yao, Y., Peng, Z., and Xiao, B.: Parallel hyper-heuristic algorithm for multi-objective route planning in a smart city. IEEE Transactions on Vehicular Technology, 67(11):10307–10318, 2018.
88. Mosadegh, H., Ghomi, S. F., and Süer, G. A.: 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.

89. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O.: Proximal policy optimization algorithms. [arXiv preprint arXiv:1707.06347](https://arxiv.org/abs/1707.06347), 2017.

VITA

SUBRAMANIAN RAMASAMY

EDUCATION	Ph.D., Mechanical Engineering, University of Illinois at Chicago, Chicago, IL, 2025. B.E., Mechanical Engineering, Sri Sivasubramaniya Nadar College of Engineering, Anna University, Chennai, TN, India, 2019.
RESEARCH	Research Assistant, University of Illinois at Chicago Jan. 2025 – Apr. 2025
TEACHING	Teaching Assistant, University of Illinois at Chicago Aug. 2024 – Dec. 2024
RESEARCH	Research Assistant, University of Illinois at Chicago Jan. 2023 – Aug. 2024
EXPERIENCE	Operations Research Intern, Amtrak May. 2022 – Dec. 2022
RESEARCH	Research Assistant, University of Illinois at Chicago Oct. 2020 – Apr. 2022
PUBLICATIONS	Journal Publications Ramasamy, S. , Reddinger, J.-P. F., Dotterweich, J. M., Childers, M. A., and Bhounsule, P. A.: Coordinated route planning of multiple fuel-constrained unmanned aerial systems with recharging on an unmanned ground vehicle for mission coverage. <i>Journal of Intelligent & Robotic Systems</i> , 106(1):118, 2022. Mondal, M. S., Ramasamy, S. , Humann, J. D., Reddinger, J.-P. F., Dotterweich, J. M., Childers, M. A., and Bhounsule, P. A.: Cooperative multi-agent planning framework for fuel constrained uav-ugv routing problem, 2023. Conference Publications Ramasamy, S. , Reddinger, J.-P. F., Dotterweich, J. M., Childers, M. A., and Bhounsule, P. A.: Cooperative route planning of multiple fuel-constrained unmanned aerial vehicles with recharging on an unmanned ground vehicle. In 2021 International Conference on Unmanned Aircraft Systems (ICUAS), pages 155164. IEEE, 2021.

Ramasamy, S., Mondal, M. S., Reddinger, J.-P. F., Dotterweich, J. M., Humann, J. D., Childers, M. A., and Bhounsule, P. A.: Heterogenous vehicle routing: comparing parameter tuning using genetic algorithm and bayesian optimization. In 2022 International Conference on Unmanned Aircraft Systems (ICUAS), pages 104113. IEEE, 2022.

Ramasamy, S., Mondal, M. S., Reddinger, J.-P. F., Dotterweich, J. M., Humann, J. D., Childers, M. A., and Bhounsule, P. A.: 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 95102. IEEE, 2023.

Ramasamy, S., Mondal, M. S., Humann, J. D., Dotterweich, J. M., Reddinger, J.-P. F., Childers, M. A., and Bhounsule, P. A.: Optimizing routes of heterogenous unmanned systems using supervised learning in a multi-agent framework: A computational study. In 2024 International Conference on Unmanned Aircraft Systems (ICUAS), pages 286294. IEEE, 2024.

Ramasamy, S., Mondal, M. S., Humann, J. D., Dotterweich, J. M., Reddinger, J.-P. F., Childers, M. A., and Bhounsule, P. A.: 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 204211. IEEE, 2024.

Mondal, M. S., **Ramasamy, S.**, and Bhounsule, P.: A bilevel optimization framework for fuel-constrained uav-ugv cooperative routing: Planning and experimental validation. arXiv preprint arXiv:2303.02315, 2023.

Mondal, M. S., **Ramasamy, S.**, Humann, J. D., Reddinger, J.-P. F., Dotterweich, J. M., Childers, M. A., and Bhounsule, P.: Optimizing fuel-constrained uav-ugv routes for large scale coverage: Bilevel planning in heterogeneous multi-agent systems.

Mondal, M. S., **Ramasamy, S.**, Humann, J. D., Dotterweich, J. M., Reddinger, J.- P. F., Childers, M. A., and Bhounsule, P.: A robust uav-ugv collaborative framework for persistent surveillance in disaster management applications. In 2024 International Conference on Unmanned Aircraft Systems (ICUAS), pages 12391246. IEEE, 2024.

Mondal, M. S., **Ramasamy, S.**, and Bhounsule, P.: An Attention-aware Deep Reinforcement Learning Framework for UAV-UGV Collaborative Route Planning. In 2024 International Conference on Intelligent Robots and Systems (IROS), pages 13687-13694. IEEE, 2024.

PRESENTATIONS

Conference Presentations

2024 IEEE International Conference on Automation Science and Engineering (CASE 2024)

2024 IEEE International Conference on Unmanned Aircraft Systems (ICUAS 2024)

2023 IEEE International Conference on Unmanned Aircraft Systems (ICUAS 2023)

2022 IEEE International Conference on Unmanned Aircraft Systems (ICUAS 2022)

2021 IEEE International Conference on Unmanned Aircraft Systems (ICUAS 2021)

Poster Presentations

2024 Midwest Robotics Workshop (MWRW 2024)

MEMBERSHIPS

IEEE Student Member