ENHANCING TEAMWORK IN MULTI-ROBOT SYSTEMS: EMBODIED INTELLIGENCE VIA MODEL- AND DATA-DRIVEN APPROACHES

A Dissertation
Presented to
The Academic Faculty

By

Esmaeil Seraj

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical & Computer Engineering
Institute for Robotics & Intelligent Machines

Georgia Institute of Technology

May 2023

© Esmaeil Seraj 2023

ENHANCING TEAMWORK IN MULTI-ROBOT SYSTEMS: EMBODIED INTELLIGENCE VIA MODEL- AND DATA-DRIVEN APPROACHES

Thesis committee:

Dr. Matthew Gombolay
School of Interactive Computing
Georgia Institute of Technology

Dr. Harish Ravichandar
School of Interactive Computing
Georgia Institute of Technology

Dr. Seth Hutchinson
School of Interactive Computing
Georgia Institute of Technology

Dr. Marynel Vázquez
Deprtment of Computer Science
Yale University

Dr. Chaouki Abdallah
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date approved: 4/24/2023

I’m not super. Any talents I have, I worked for – it seems a long time since I thought of myself as a hero.

Oliver Queen

For Mom, Dad, Binazir, Reza, Reyhaneh, Maryam, Helma, and Ardavan..!

ACKNOWLEDGMENTS

I would like to thank the members of my thesis committee for their help and guidance in preparation of this work and for their support throughout these years.

Special thanks are due to my advisor, Dr. Matthew Gombolay without whom completion of this work would have been impossible. I am grateful for his endless support and mentoriship. Thank you!

I would like to thank my academic advisor, Dr. Daniela Staiculescu of ECE for all her kindness, friendship, support, and guidance since before I was admitted to Georgia Tech!

I would also like to thank my friends and family, Reza, Reyhaneh, Maryam, Helma, Hamid, Hoda, Behnam, Ali, Elnaz, and Ardavan as well as my work friends and friends in the Lab (what can I say? I have a lot of friends!), Rohan, Zac, Vahid, Mariah, Andrew, Manisha, and everyone else whom I've interacted with during these years around here. You guys helped my stay sane during this process. You guys made it possible.

Finally, nobody has been more important to me in the pursuit of this journey than my family. I'd like to thank my beautiful wife, my best friend, and the love of my life, Binazir, for all her love and support. You have always been there for me and thank you for making everything worth it! and lastly, I want to thank my parents, whose love and support I've always been blessed with, regardless of where I was and what I did. Mom, you're my rock, you're my hero.

The author gratefully acknowledges the supports by the Naval Research Lab under grant N00173-21-1-G009, Lockheed Martin Corporation under grant GR0- 0000509, and Office of Naval Research (ONR) under grant N00014-19-1-2076 for their support throughout the PhD period.

TABLE OF CONTENTS

Acknowledgments v

List of Tables xiii

List of Figures xv

List of Acronyms xxiv

Summary xxvi

Chapter 1: Introduction and Literature Survey 1

1.1 Motivation 1

1.2 Thesis Contributions 2

Chapter 2: Literature Survey 6

2.1 Model-Based Multi-agent Planning and Control Methods 6

2.1.1 Dynamic Field Coverage and Mobile Sensor Networks 6

2.1.2 Heterogeneous Multi-Robot Teams 10

2.2 Learning End-to-End Multi-agent Coordination Policies 11

2.2.1 Enabling MARL with Communication 11

2.2.2 Modeling Heterogeneous Multi-Agent Systems via MARL 13

2.2.3 Decision Rationalization and Mutual Information (MI) in MARL . . 14

2.2.4 Multi-Agent Learning from Expert Demonstration (MA-LfD) . . . 15

Chapter 3: Preliminaries and Background 18

3.1 Motivating Applications 18

3.1.1 Aerial Wildfire Monitoring 18

3.1.2 Robotic Teams for Wildfire Fighting 20

3.2 FARSITE Fire Propagation Mathematical Model 22

3.3 Markov Decision Processes and Reinforcement Learning 24

3.3.1 Policy Gradient Methods 25

3.4 Multi-Agent Reinforcement Learning 25

3.5 Learning from Demonstration 27

Chapter 4: Model-Based Node-Level Control for Multi-Robot Coordination . . 30

4.1 Coordinated Control of UAVs for Human-Centered Active Sensing of Wildfires 31

4.1.1 Introduction and Motivation 31

4.1.2 Preliminaries: Adaptive Extended Kalman Filter (AEKF) 34

4.1.3 Problem Statement and Algorithmic Overview 34

4.1.4 Method 37

4.1.5 Online Inference of Wildfire Dynamics 40

4.1.6 Controller Design 44

4.1.7 Empirical Evaluation 48

4.1.8 Conclusion 51

4.2 Adaptive Leader-Follower Control for Multi-Robot Teams with Uncertain Network Structure 53

4.2.1 Introduction and Motivation 53

4.2.2 Problem Statement and Formulation 56

4.2.3 Adaptive Leader-Follower Controller 57

4.2.4 Empirical Evaluation 65

4.2.5 Demonstration: Multi-Robot Testbed 68

4.2.6 Conclusion 70

Chapter 5: High-Level Performance-Guaranteed Coordinated Planning for Multi-Robot Teams 71

5.1 Multi-UAV Planning for Cooperative Dynamic Field Coverage with Quality-of-Service Guarantees 72

5.1.1 Introduction 73

5.1.2 Related Work 77

5.1.3 Problem Formulation and Algorithmic Overview 79

5.1.4 Coordinated Planning for Monitoring with Guaranteed Quality-of-Service 84

5.1.5 Coordinated Distributed Field Coverage Module (No Specified Areas) 97

5.1.6 Empirical Evaluation 99

5.1.7 Demonstration: Multi-robot Testbed 108

5.1.8 Limitations and Future Work 111

5.1.9 Conclusion 112

5.2 A Hierarchical Coordination Framework for Joint Perception-Action Tasks in Composite Robot Teams 114

5.2.1 Introduction and Motivation 114

5.2.2 Problem statement and Formulation 118

5.2.3 High-level Decision-Making 123

5.2.4 Low-level Coordinated Control and Planning 133

5.2.5 Empirical Evaluation 147

5.2.6 Demonstration: Multi-Robot Testbed 151

5.2.7 Discussion 153

5.2.8 Conclusion 156

Chapter 6: Learning End-to-End Multi-agent Coordination and Communication Policies 157

6.1 Learning Efficient Communication for Cooperative Heterogeneous Teaming 157

6.1.1 Introduction and Motivation 158

6.1.2 Related Work 160

6.1.3 Preliminaries 162

6.1.4 Method 164

6.1.5 Training and Execution 171

6.1.6 Empirical Evaluation 172

6.1.7 Conclusion 179

Chapter 7: Iterative Reasoning for Multi-Agent Decision-Making Under Uncertainty 181

7.1 Iterated Reasoning with Mutual Information in Cooperative and Byzantine Decentralized Teaming 181

7.1.1 Introduction and Motivation 182

7.1.2 Related Work 184

7.1.3 Preliminaries 185

7.1.4 Mutual Information Maximizing Policy Gradient 187

7.1.5 Empirical Evaluation 195

7.1.6 Results, Ablation Studies, and Discussion 197

7.1.7 Limitations and Future Work 199

7.1.8 Conclusion 200

Chapter 8: Learning Heterogeneous Teaming from Human Demonstration . . . 201

8.1 Mixed-Initiative Multi-Agent Apprenticeship Learning for Human Training of Multi-Robot Teams 201

8.1.1 Introduction 202

8.1.2 Related Work 206

8.1.3 Problem Formulation: General MA-LfD with Heterogeneous Agents 207

8.1.4 Mixed-Initiative Multi-Agent Apprenticeship Learning (MixTURE) 209

8.1.5 Evaluation 215

8.1.6 Conclusion 224

Chapter 9: Limitations and Future Work 225

Chapter 10:Conclusion 227

Appendices 231

Chapter A:Time Independency of the EKF's Measurement Residual 232

Chapter B: Calculating Coordinates of Reachable Polygon's Vertices in Scanning Framework 237

Chapter C:Supplementary Information (chapter 6) 239

C.1 Evaluation Environment: Additional Details and Parameters 239

C.1.1 Predator-Prey (PP) [251] 239

C.1.2 Predator-Capture-Prey (PCP) 240

C.1.3 FireCommander (FC) [54] 240

C.2 Supplementary Results and Ablation Studies 241

C.2.1 Implementation and Model Details 242

C.2.2 Message Binarization Method: An Ablation Study 243

Chapter D: Supplementary Information (chapter 7) 244

D.1 InfoPG Pseudocode 244

D.2 InfoPG for Continuous Action-Spaces 246

D.3 InfoPG Objective Function Derivation (Equation 7.2) 247

D.4 Convergence Proof Sketch for Equation 7.2 247

D.5 Full Proof of Theorem Theorem 5 252

D.6 Discussion on the Uniformity of Priors Assumption 254

D.7 Supplementary Results 254

D.7.1 Policy Interpretation for the Fraudulent Agent Experiment 255

D.7.2 A Qualitative Analysis for Bounded Rationality 258

D.7.3 Scalability Analysis: Pistonball 261

D.7.4 Agent-wise Performance Comparison 263

D.8 Evaluation Environments: Additional Details and Parameters 263

D.9 Training and Execution Details 266

Chapter E: Supplementary Information (chapter 8) 272

E.1 Algorithm Details and Pseudocode 272

E.2 Proof of Evidence Lower-Bound (ELBO) in Eq. 3 273

E.3 Environment Details 273

E.3.1 Predator-Prey (PP) 273

E.3.2 Predator-Capture-Prey (PCP) 273

E.3.3 FireCommander (FC) 275

E.4 Human-Subject User Study Details: Environment and Procedures 276

E.4.1 General Game Objectives and Logistics 276

E.4.2 Understanding the Game Score Function 278

E.4.3 Task Description for Each Condition 278

E.5 Expert Heuristic Design Details 281

E.5.1 Predator-Prey (PP) Expert Heuristic 281

E.5.2 Predator-Capture-Prey (PCP) Expert Heuristic 282

E.5.3 FireCommander (FC) Expert Heuristic 282

E.5.4 Communication Heuristic 284

E.6 Ablation Studies and Supplementary Results 284

E.6.1 Scalability 284

E.7 Hyperparameters 284

References 286

LIST OF TABLES

5.1 Summary of key nomenclature used in our paper. 80

5.2 Summary of key nomenclature used in our paper. 124

5.3 Results of our Sensitivity experiment for different values of agents' FOV sizes. Training and predictions times unit is $[\frac{Sec.}{Ep.}]$. 151

5.4 Results of our performance sensitivity evaluation under horizontal sweeping, diagonal sweeping and spiral exploration paths. Training and predictions times unit is $[\frac{Sec.}{Ep.}]$. 151

6.1 Reported results are Mean ($\pm$ Standard Error (SE)) from 50 evaluation trials. For all tests, the final training policy at convergence is used for each method. As shown, HetNet outperforms all baselines in all three domains. 174

7.1 Reported results are Mean (Standard Error) from 100 testing trials. For all tests, the final training policy at convergence is used for each method and for Informational Policy Gradient (InfoPG) and Adv. InfoPG, the best level of k is chosen. 198

8.1 Full evaluation results for MixTURE and baselines in all domains and all difficulty levels on expert heuristic dataset. Heu. and Diff. indicate the models access to heuristic or differentiable communication, respectively. Bolded values indicate the best models. 221

D.1 Co-op Pong Training Hyperparameters. 268

D.2 Pistonball Training Hyperparameters. 269

D.3 Multiwalker Training Hyperparameters. 269

D.4 StarCraft II Mini-game (The 3M Challenge) Training Hyperparameters. . . 269

D.5 Pistonball Training Hyperparameters for the fraudulent agent experiment. . 271

E.1 Environment Configuration Details 274

E.2 Expert Heuristic Performance (# steps) 283

E.3 Hyperparameters 285

LIST OF FIGURES

3.1 Figure 3.1a demonstrates an example of separating wildfire areas for coverage and tracking based on fire propagation velocity and direction. The firefront in each area is moving in a different direction and with a varying velocity and, therefore, at least one separate UAV is required to monitor and track the firefront in each area. In a similar perspective, Figure 3.1b shows an example of disjoint coverage areas that are separated according to areas of human (i.e., firefighters) activity and priority. In this case, the autonomous UAVs are teaming with humans to safely manage their human collaborators by providing for them *high-quality* information regarding their time-varying proximity to fire (i.e., areas of human operation). When we have access to additional UAV resources that may not be needed in the prioritized areas, we can deploy such UAVs (i.e., unallocated UAVs) to monitor the rest of the wildfire areas (not specified or prioritized). 19

3.2 Figure Figure 3.2a depicts a firefighting Airtanker dropping fire retardant on the 2008 Tea Fire, Montecito, Calif. (Courtesy: National Public Radio, July 2012). Figure Figure 3.2b depicts a composite UAV team for aerial wildfire fighting. Smaller quadcopters (perception agents) sense the fire and gather information for larger, fixed-wing fire-extinguisher aircraft, i.e. action agents (Courtesy: Dronelife, Feb. 2016). 21

4.1 Firefighters trying to control a back burn as the Carr fire spreads toward Douglas City. Courtesy of the Los Angeles Times (May 2019). 32

4.2 Fusing AEKF measurement residual and human GPS data to generate an uncertainty map. UAVs focus on areas of human activity while monitoring fire propagation. 39

4.3 The UAV observation model. 42

4.4 Node-level controller architecture of each drone d. 45

4.5 This figure depicts a quantitative comparison of our coordinated controller with prior work (left-side) and human safety index (SI) variations as a temporal quantity, with respect to distance between an approaching firefront and a human firefighter (right-side). 49

4.6 Simulation results of eight sample time-steps between $t = 20$ (top-left) and $t = 300$ (bottom-right) for distributed coverage, representing drone FOVs projected on the ground. Dot rays show the FOV centroids. 51

4.7 Feasibility of the proposed distributed control algorithm for wildfire coverage evaluated on physical robots in Robotarium platform [151]. The experiment footage can be found at `https://youtu.be/j3YdIO5u_fE`. 52

4.8 This figure depicts the empirical stability and convergence results of our coordinated adaptive leader-follower controller with network disconnectivity. Figure 4.8a presents the error convergence and the control effort for each of the ten robot agents in the team (setup described in subsubsection 4.2.4), and Figure 4.8b shows the adaptive gains during the simulation. 66

4.9 This figure demonstrates the follower agents' behaviors under our proposed coordinated adaptive framework in comparison with the conventional controller. The black and blue lines show the positions of followers and leaders, respectively. 67

4.10 Effects of choosing reference communication graph model on the performance of the leader-follower controller and follower agents' behaviors (e.g., position and convergence) for complete (Figure 4.10a), cycle (Figure 4.10b) and linear (Figure 4.10c) graphs. 68

4.11 Initial ($t = 0s$) and final robot standings ($t = 80s$) in the field coverage experiment under both the conventional (left-side column) and our proposed controllers (right-side column). In our experiments, three disjoint groups (no communication between groups) of two, two, and three robots, were initialized. A cycle graph was selected as the reference communication model with leader locations set on lower-left and upper-right corners of the designated coverage area. (video link `https://youtu.be/mX8Sm-iH-kQ`). . . . 69

5.1 A flowchart diagram representing the URR checking process as the analytical safety condition. If the URR bound is not satisfied for a node in g_t, the graph is partitioned into two sub-graphs g_t^1 and g_t^2 (i.e., through k-means clustering), and accordingly into two paths, and another UAV is called-in from the available UAVs in the team. The process is repeated until URR is satisfied for all sub-graphs. 82

5.2 The three considered scenarios in which, a fire (i.e., target) could be stationary (left), moving (middle) or moving-and-spreading (right). Green dots show a firespot's current location and red circles represent the distribution over the firespot's next-step location. Blue circles show the Steiner zones. Note that the black, solid and dashed lines represent a UAV's current and next-step tour paths, respectively. 90

5.3 The left-side figure depicts a quantitative evaluation of our analytical URR bound and shows that increasing the number of disjoint areas to be monitored resulted in a rise in the number of UAVs required to guarantee the performance. This also held as the wildfire propagation scenario changed and a fire propagated more aggressively. The right-side figure demonstrates the efficacy of the coordinated, distributed coverage algorithm, in which less cumulative uncertainty residuals (Y-axis) is better. This figure shows that UAVs easily covered the fire map in the case of stationary wildfire, while in the case of a fast-growing fire, more UAVs were required to cover the fire map efficiently. 101

5.4 The computed upper-bound times, T_{UB}, it takes a UAV to complete a determined tour over the course of a simulation with respect to the three wildfire scenarios. As shown, the computed upper-bound service times initially start with a high value, relative to the wildfire scenario, and gradually decrease until converging to a reasonable value, as the algorithm determines that more UAVs are required for a guaranteed tracking of the firespots. Plots are averaged across UAVs and across ten separate experiment trials. 103

5.5 Figure 5.5a shows the $\frac{T_{UB}}{T^*}$ ratio for all fire scenarios computed over 5000 trials for each case. This ratio evaluates the sanity of the derived upper-bound times, T_{UB}, and the actual maximum time it takes UAVs to complete a tour, T^*. As shown, for a tight upper-bound, the value of this ratio must remain close to one. Figure 5.5b shows the sum of all UAVs' uncertainties for all of the measurements, averaged over time. As shown, the measurement uncertain residuals are bounded. Figure 5.5c compares the measurement uncertainty residuals for our approach and [36] in the evolving fire scenario (i.e., fire starts in Case 1 and evolves into Case 2 and 3 at certain time-steps). As shown, our method outperforms the baseline in this more challenging scenario. Figure 5.5d shows the combined uncertainty maps generated by summing all measurement residuals of all UAVs for all firespots and averaging over time. 105

5.6 This figure presents a comparison of our predictive coordinated controller to prior works for both our fire environment (bottom row) and that of [119] (top row). First column figures show the cumulative uncertainty residual during 100 simulation time steps (10 trials). Middle column figures show the uncertainty residual at the end of last simulation step over the course of 2500 episodes of 100 time-step simulation. The third column figures show the RL agent's loss over the course of training. 106

5.7 This figure presents example demonstrations of the proposed coordinated planning algorithm for human safety, implemented on physical robots. In Figure (9.4) all URRs are satisfied and the algorithm determines that a team of three robots can collectively provide the estimated information for the three areas without losing the track of any spot. Link to video: `https://youtu.be/zTR07cKlwRw`. 109

5.8 This figure presents four sample demonstrations of our experiments for the proposed coordinated field coverage algorithm, implemented on physical robots. Each sub-figure 10.1 – 10.4 includes two images: (1) moderate moving fire (left-side) and (2) fast moving-spreading fire (right-side). Figure 10.1 represents the initial step in the algorithm in which *unallocated* robots are selected. Figure 10.4 shows that the robot team can successfully cover both wildfire cases and monitor the dynamic spots, after ten minutes of running the algorithm. Link to video: `https://youtu.be/zTR07cKlwRw` . . 110

5.9 This figure depicts the proposed hierarchical coordination framework. Dashed links pass control input u_t to and receive state information $\hat{q}_t$ from robots. . 120

5.10 This figure depicts the designed, landscape-based, action and state space representation in the multi-agent Semi-MDP (middle). Our action-space (right side) can be represented as $\mathscr{A} \in \{0,1\}^{w_1 \times w_2}$, where an entry $\mathscr{A}_{i,j} = 1$ indicates the location of an *exploitation* action's target, and when $\mathscr{A}_{i,j} = 0, \forall i, j \in \mathbb{N}$ an *exploration* action is declared by the decision-making module. We take a similar approach to represent the state-space (left side) as overlaid feature maps forming a feature tensor $\mathscr{S} \in \{S_t\}^{w_1 \times w_2 \times k}$. 128

5.11 This figure depicts an example of timings and transitions in our discrete-event simulation of continuous time progression. The top gray panel represents the transitions in agent 1's perspective. 130

5.12 This figure depicts the time-dependency of measurement uncertainty as propagated by EKF. The dropping uncertainty regardless of the time of visit will always have the same value ($\varepsilon^q_{t_2} = \varepsilon^q_{t_3}$), if the observing robot's displacement to target is similar at both times. 136

5.13 Perception agent accounts for the assigned manipulator agent's motion model, including its maximum turning rate ω^M_{max} (dashed black lines) and minimum and maximum linear velocities $\left[v^M_{min}, v^M_{max}\right]$ (red dashed lines). Sensing agent only scans the reachable areas for targets (green dots), enclosed by four vertices (1)-(4), (black dots). 140

5.14 This figure depicts collaborative planning for coordinating between perception and action agents (UAVs in this case). Figure 5.14a and Figure 5.14b represent the stationary and dynamic target scenarios, respectively. Note that the moving targets in the second case (black arrows) in Figure 5.14b, lead to a delayed exploitation by the perception UAV. 144

5.15 This figure depicts our benchmark comparison results. The leftmost figure shows a comparison on reward between our approach and benchmarks. Test repeated for 100 times. Error bars are standard error. The middle chart depicts a comparison on time (unit time in simulation) taken to find all fires and put out all fires. Test repeated for 100 times. Error bars are standard error. The rightmost figure shows our RL agent's training reward over the course of training averaged over three runs, where the shaded region depicts the standard deviation. 147

5.16 This figure depicts our scalability and computation time results. The leftmost figure shows the mean reward accumulated in nine different scenarios specified by the number of agents in the composite team and the number of firespots in the environment. Test repeated for 10 times and standard deviations are presented ($\pm$ STD). The middle and rightmost charts depict the training and prediction times for the same nine scenarios, respectively. The times are shown as seconds per episodes. 150

5.17 (1) Two perception robots start exploring the environment for targets. (2) Perception robots evaluate the Tracking Error Bound (TEB) bound for an action robot. (3) The action robot starts the task the first time TEB is violated; perception robots continue to explore. (4) Final standing of the robot team after servicing all six initially unknown targets. 153

5.18 This figure presents four sample demonstrations of our coordinated control and planning framework implemented on physical robots (initial to final standings) in the complex scenario (B) described in subsubsection 5.2.6 (`https://youtu.be/qcomKuD-Hhw`). 154

6.1 Overview of our multi-agent heterogeneous attentional communication architecture in a Centralized Training and Decentralized Execution (CTDE) paradigm. At each time point $t = t_0$, each agent j of class i generates a local embedding from its own inputs, by passing its input data through class-specific preprocessing units (i.e., a Convolutional Neural Network (CNN) or a fully-connected Neural Network (NN)) and a Long Short-Term Memory (LSTM) cell. The embeddings are then passed into the Heterogeneous Graph-Attention (HetGAT) communication channels by each agent where a Gumbel-Softmax process is used in a class-specific encoder network to convert the message into binarized messages, m_t^{jk},. The message is then sent to neighboring agents where a class-specific decoder is used to decode all the received messages, aggregate them and weight them according to some learned attention coefficients. The aggregated message information is leveraged by the receiving agent to compute the action probabilities as its policy output. These decentralized per-class policies are then used for action decisions of different agents in each class. 165

6.2 The sender and receiver phases of the feature update process in a HetGAT layer for one agent, j, of class i. 169

6.3 Average steps taken (± SE) by each method across episodes and three different random seeds as training proceeds. HetNet outperforms all baselines in both domains. 176

6.4 Communicated bits per round of communication vs. performance in PCP for different methods. HetNet facilitates binarized messages among agents which requires significantly less CB as compared to real-valued baselines. . 177

6.5 Analyzing Heterogeneous Policy Networks (HetNet)'s performance with and without communication (Figure 6.5a) and across different binary message dimensions (Figure 6.5b) in the PCP domain. Communication policy learned by HetNet improves the cooperativity among agents and the performance improves with larger message sizes. Figure 6.5c depicts results for analyzing HetNet's ability to scale to different number of agents. As shown, HetNet-Binary can successfully scale to different sizes of the composite team. . . . 178

6.6 Learning curves during training as well as the test results (average number of steps taken) for final policies learned by centralized, per-class and per-agent critic architectures in the PCP domain. 179

7.1 An instance of the information flow in a k-level decision hierarchy between two agents i and j for calculating their level-k strategies. Level-zero actions (e.g., $a^{i,(0)}$) represent the naive non-strategic actions. 188

7.2 (Top Row) Team rewards obtained across episodes as training proceeds. The shaded regions represent standard error. Our Advantage InfoPG (Adv. InfoPG) continually outperforms all baselines across all domains and in both training and testing (see Table 7.1). (Bottom Row) The MI ablation study results, comparing the MI variations between InfoPG and MOA (MI-based baseline) where InfoPG demonstrates a higher final average MI estimate across all domains. The shaded blue region represents the area between InfoPG's lower and upper bounds on MI. 196

7.3 The fraudulent agent experiment scenario (Figure 7.3a) and training results (Figure 7.3b) in the Pistonball domain, comparing the team reward performance for Adv. InfoPG (Equation 7.4), InfoPG (Eq.Equation 7.3) and MOA. 199

8.1 The proposed MixTURE architecture for human training of robot teams with MIM-based learned differentiable inter-agent communication for a two-agent example scenario. At each timestep, each agent generates an embedding from its local observation, which are then passed into local recurrent policies (to handle partial observability) for each agent. To learn from the expert demonstrations stored in the dataset D_E, we build a distributed multi-agent GAIL architecture (i.e., per agent discriminators, $\mathscr{D}_\theta$). We enable inter-agent communication by adding an attentional communication module enhanced by a MI maximization reverse model. 211

8.2 Full evaluation results for MixTURE and the baselines in the all difficult levels (i.e., easy, moderate, and hard) of the three environments (i.e., PP, PCP, and FC). 217

8.3 Evaluation results for MixTURE and MA-GAIL [110] on real human data. From left to right, the figures present results for easy, moderate, and hard scenarios (see Section subsubsection 8.1.5). MixTURE can learn high-quality multi-agent policies from human demonstration. 220

8.5 Statistical analysis for the human subject data, supporting the **H3**. Results indicate that using the MixTURE model w/o communication demonstration leads to significantly higher performance, more tasks completed and a better ability to scale to the more complex scenarios with more tasks, and significantly lower demonstration time per step. 223

8.4 Workload and SUS measures, compared for against the experiment conditions.223

B.1 Vertex coordinates of the reachable polygon, introduced in Figure 5.13, are calculated to generate fixed-wing-friendly paths. 238

C.1 Performance comparison for two different binarization methods: (1) STE and (2) Gumbel-Softmax, for 8-bits and 16-bits message dimensions. 242

D.1 Action distributions of piston agents across 37 timesteps for the fraudulent agent experiment introduced in section 7.1. Note that Piston 2 (not displayed) is the fraudulent agent with untrainable random policy. 256

D.2 Comparing the learned policies by InfoPG at convergence in the SC2 domain with $k = 0$ and $k = 1$. With $k = 0$ (Fig. 5.1–5.3), the naive agents disregard other agents actions and simply learn to run away to avoid the negative penalties of getting shot at. This is while the more sophisticated agents with $k = 1$ (Fig. 5.4–5.6), learn more strategic policies to work together to eliminate the enemy team and achieve high positive rewards. 259

D.3 Comparing the learned policies by InfoPG in the Multiwalker with $k = 1$, $k = 2$, and $k = 3$, 5a-5c, respectively. With $k = 1$, agents only learn to perform a split to balance the package on top and avoid falling. This is while with $k = 3$ agents learn to quickly walk forward. The middle stage of rationalization, $k = 2$, achieves an in-between policy where agents split to balance, but also wiggle forward slowly. 260

D.4 Scalability comparison between Adv. InfoPG and the best-performing baseline, MOA [99], in the Pistonball domain with ten interacting agents. Adv. InfoPG outperforms MOA in both maximizing average individual and team reward performances. 262

D.5 Instances of the utilized multi-agent cooperative environments. Domains are parts of the PettingZoo [282] MARL research library and can be accessed online at `https://www.pettingzoo.ml/envs`. The StarCraft II [284], can be accessed from Deepmind's repository available online at `https://github.com/deepmind/pysc2`. 263

D.6 Individual rewards obtained by each individual agents across episodes as training proceeds in the three evaluation environments. Our Adv. InfoPG continually outperforms all baselines [100, 99, 63, 126] across all domains. 270

E.1 The PP, PCP, and FC domains utilized for evaluating MixTURE and the baselines. 274

E.2 Designed FireCommander interface for the human-subject user experiment. 276

E.3 Instances of the designed FireCommander game environment at different levels for the human-subject user experiment. 279

E.4 The communication menu designed for expert demonstration in the second condition (i.e., withComm mode). The messages in the communication menu designed for this condition include a wide range of options from anticipatory (i.e., sharing information by anticipating one another's needs rather than answering to requests or prompts) and deliberative information sharing (i.e., prioritizes information about the next goal to be accomplished during a task), the design of which was inspired by prior work [309, 310]. The dimensionality of this message-space is 26. 282

E.5 Full evaluation results for MixTURE and the baselines in the all difficult levels (i.e., easy, moderate, and hard) of the three environments (i.e., PP, PCP, and FC). 283

SUMMARY

High-performing human teams leverage intelligent and efficient communication and coordination strategies to collaboratively maximize their joint utility. Inspired by teaming behaviors among humans, I seek to develop computational methods for synthesizing intelligent communication and coordination strategies for collaborative multi-robot systems. I leverage both classical model-based control and planning approaches as well as data-driven methods such as Multi-Agent Reinforcement Learning (MARL) and Learning from Demonstration (LfD) to provide several contributions towards enabling emergent cooperative teaming behavior across robot teams.

In my thesis, I first leverage model-based methods for coordinated control and planning under uncertainty for multi-robot systems to study and develop techniques for efficiently incorporating environment models in multi-robot planning and decision making. In these contributions, I design centralized and decentralized coordination frameworks, at the control-input and the high-level planning stages, which are informed by and have access to the model of the world. First, I develop an algorithm for *human-centered* coordinated control of multi-robot networked systems in safety-critical applications. I tackle the problems of enabling a robot team to reason about a coordinated coverage plan through active state estimation and providing probabilistic guarantees for performance. I then extended these methods to directly formulate and account for heterogeneity in robots' characteristics and capabilities. I design a hierarchical coordination framework, which enables a composite team of robots (i.e., including robots that can only sense and robots that can only manipulate the environment) to effectively collaborate on complex missions such as aerial wildfire fighting.

Model-based approaches provide the ability to derive performance and stability guarantees. However, can be sensitive to the accuracy of the model and the quality of the

heuristic algorithm. As such, I leverage data-driven and Machine Learning (ML) approaches, such as MARL, to provide several contributions towards learning emergent cooperative behaviors. I design a graph-based architecture to learn efficient and diverse communication models for coordinating cooperative heterogeneous teams. Finally, inspired by the theory of mind in humans' strategic decision-making model, I develop an iterative model to learn deep decision-rationalization for optimizing action selection in collaborative, decentralized teaming.

In recent years, MARL has been predominantly used by researchers to optimize a reward signal and learning multi-robot tasks. Nevertheless, Reinforcement Learning (RL) generally suffers from key limitations such as the difficulty of designing an expressive and suitable reward function for complex tasks, and high sample complexity. As such, accurate models of human strategies and behaviors are increasingly important. Additionally, as multi-robot systems become increasingly prevalent in our communities and workplace, aligning the values motivating robot behaviors with human values is critical. LfD attempts to learn the correct behavior directly from expert-generated data demonstrations rather than a reward function. As such, in the last part of my work, I develop a multi-agent LfD framework to efficiently incorporate humans' domain-knowledge of teaming strategies for collaborative robot teams and directly learn team coordination policies from human teachers. To this end, I propose Mixed-Initiative Multi-Agent Apprenticeship Learning (MixTURE) framework for human training of robot teams. MixTURE enables robot teams to learn a humans' *preferred* strategy to collaborate, while simultaneously learning end-to-end emergent communication for the robot team to efficiently coordinate their actions, without the need for human generated data. MixTURE benefits from the merits of LfD methods over RL, while significantly alleviating the human demonstrators workload and time required to provide demonstrations, as well as increasing the System Usability Scale (SUS) and overall collaboration performance of the robot team.

CHAPTER 1
INTRODUCTION AND LITERATURE SURVEY

1.1 Motivation

While multi-robot systems are capable of executing time-sensitive, complex, and large-scale problems, it is challenging to efficiently coordinate such systems and to optimize the collaborative behavior among robots. Communication is a key necessity to achieve an effective coordinated policy among agents. This process, in fact, emulates high-performing human teams where communication is leveraged to build team cognition and maintain shared mental models to improve team effectiveness [1]. To achieve such capabilities in multi-robot systems, we require algorithms that enable *collective intelligence* in robots for planning and decision-making under uncertainty such that their shared utility is maximized. Collective intelligence in multi-robot systems refers to the ability of a group of robots to work together and achieve a goal that is beyond the capability of an individual robot. It involves combining the individual knowledge and skills of each robot to create a more sophisticated and intelligent system as a whole [2, 3].

From a control theoretic perspective, multi-robot coordination approaches can be tackled at two separate levels: (1) high-level decision-making and (2) low-level control [4]. The high-level decision-making module usually attempts to address the questions of "*what to do?*" or "*where to go?*" and thus, deals with planning a set of objectives (course of actions) among several possible options through which robot(s) can optimally (or at least satisfactorily) accomplish their task-objective. On the other hand, the low-level control module tackles the question of "*how to do it?*" or "*how to go there?*" and addresses the problem of designing appropriate control inputs for robot actuator(s) so that the robot(s) can follow a future trajectory as closely as possible. A hierarchical control architecture consists of both aforementioned low- and high-level modules and usually solves these two modules consecutively, such that the low-level module generates the control inputs required for the robot(s) to execute the plan made by the high-level decision-maker [4].

Low-level controllers or high-level policies and plans can be synthesized based on models (of the world or agents) or learned end-to-end through data (interaction or demonstrated). Regardless of the approach used, however, the goal of creating a robot *team* is to build a group of coordinated agents, empowered via collective intelligence and social dexterity among themselves, that are capable of true, shoulder-to-shoulder collaboration, rather than simply co-existing [1]. In this thesis, I seek to develop such computational methods for synthesizing intelligent communication and coordination strategies for collaborative multi-robot systems. I leverage both classical model-based control and planning approaches as well as data-driven methods such as MARL and Multi-Agent Learning from Demonstration (MA-LfD) to provide several contributions (see section 1.2 for a summary of contributions) towards enabling emergent cooperative teaming behavior in multi-robot systems.

In tackling the problem of designing or learning multi-agent coordination strategies through end-to-end models, I specifically focus my studies around three major problems that are less addressed in prior work: (1) heterogeneous teaming, (2) complex multi-faceted objective(s), and (3) *restless* (i.e., changing states regardless of robot actions) and dynamic environments. In first two chapters (chapter 4-chapter 5), I motivate the multi-robot coordination and planning problem in the dynamic field-coverage application via robot teams. Later, in chapter 6-chapter 8, I move beyond the application of multi-robot systems in collaborative field coverage and investigate robot coordination strategies and decision-making in a variety of domains and applications.

1.2 Thesis Contributions

The central statement and claim underlaying my works in this thesis is as follows: **Enabling robot teams to efficiently communicate and to reason about their plans, policies, and**

actions-decisions will improve the collective team coordination and collaboration performance. I support this claim through the following contributions:

- **Developing a novel coordinated control framework for UAV teams to enable human-centered active sensing of dynamic environments**: I develop a dual-criterion objective function based on Kalman uncertainty residual propagation and weighted multi-agent networked control, which enables the UAVs to actively infer a wildfire's propagation model parameters and monitor the fire transitions [5].

- **Developing an adaptive control architecture for dealing with uncertain and faulty communication networks in multi-robot teams**: I develop a centralized, coordinated-control structure for multi-robot teams with uncertain network structure achieved through a model-reference adaptive control architecture. This approach enables multi-robot teams to achieve consensus even with *disconnected* communication graphs [6].

- **Creating a multi-agent coordinated planning for cooperative dynamic field Coverage with quality-of-service guarantees**: Motivated by the problem of aerial wildfire monitoring, I propose a predictive framework which enables cooperation in multi-UAV teams towards collaborative dynamic field coverage with probabilistic performance guarantees. This approach enables UAVs to infer the latent fire propagation dynamics for time-extended coordination in safety-critical conditions [7, 8].

- **Developing a hierarchical coordination framework for heterogeneous, collaborative robot teams**: I introduce a Multi-Agent State-Action-Reward-Time-State-Action (MA-SARTSA) algorithm under Multi-Agent Partially Observable Semi-Markov Decision Process (MA-POSMDP) formulation as a multi-agent decision-making framework to enable agents to learn to surveil in an environment with an unknown number of dynamic targets [9]. MA-SARTSA tackles an *asynchronous* multi-agent decision-making process with macro action.

- **Introducing an MARL framework to learn highly efficient and diverse communication models for coordinating cooperative heterogeneous teams**: I introduce Heterogeneous Policy Networks (HetNet) through which different classes of robots can learn from scratch to "speak" in binary as a highly-efficient intermediate shared language among agents and collaborate [10].

- **Developing a MARL framework for decision-making under bounded rationality**: I introduce Informational Policy Gradient (InfoPG), which is Inspired by the *k*-level reasoning from the cognitive hierarchy theory [11] and strategic decision-making in humans. InfoPG enables iterated decision rationalization for cooperative MARL under the assumption of bounded rational agents [12].

- **Developing a MA-LfD framework to efficiently incorporate humans' domain-knowledge of teaming strategies for collaborative robot teams and directly learn team coordination policies from human expert teachers**: I propose MixTURE framework for human training of robot teams. MixTURE enables robot teams to learn a humans' *preferred* strategy to collaborate, while simultaneously learning end-to-end emergent communication for the robot team to efficiently coordinate their actions, without the need for human generated data.

The remaining of this thesis is organized as follows: chapter 2 presents a through literature survey of the related prior works. In chapter 3, preliminaries and background information are provided to help motivate the thesis and review a few fundamental concepts required in later chapters. Chapter 3 presents two novel model-based approaches ([5] and [6]) for multi-robot coordination at the node level with applications to dynamic field coverage via UAV agents. Chapter 4 extends the works in chapter 4 by considering safety-critical and time-sensitive scenarios where only a limited number of UAVs are available for allocation or the UAV team consists of robots with heterogeneous capabilities and tasks. This chapter presents a performance-guaranteed model-predictive approach [7, 8] and a data-based solution [9] to plan coordinated policies for robot teams under environment uncertainty. In chapter 6 a novel method for learning highly-efficient end-to-end multi-agent coordination and communication policies is presented [13, 10]. Chapter 6 presents an enhancement over previous data-based and ML solutions for multi-robot coordination by a novel MARL architecture that enables iterated reasoning and decision-rationalization for agents of a cooperative multi-robot team [12]. Lastly, chapter 8 presents a novel MA-LfD framework to efficiently incorporate humans' domain-knowledge for collaborative robot teams and directly learn team coordination policies from human teachers.

CHAPTER 2
LITERATURE SURVEY

This chapter presents a comprehensive literature survey on related prior work in two subcategories: (1) recent model-based multi-agent planning and control methods, and (2) learning end-to-end multi-agent coordination and collaboration policies.

2.1 Model-Based Multi-agent Planning and Control Methods

2.1.1 Dynamic Field Coverage and Mobile Sensor Networks

Recent advances in UAV technology have opened up the possibility of providing real-time, high-quality fire information to firefighting teams across thousands of acres [14, 15, 16]. However, coordinated control of UAV teams in this volatile setting provides particular challenges, such as decentralized controller design, task scheduling problems, large-scale communication, smoke detection and image stabilization, etc. [17, 18, 19, 20, 21].

Among the vision-based fire coverage methods [16, 22, 23, 24], Merino *et al.* [25] investigated the utility of visual and infrared cameras on UAVs to monitor the evolution of the firefront shape and then attempted to extend the proposed approach to a collaborative scheme performed by multiple UAVs. However, Yuan *et al.* [26] note that vision-based approaches for fire monitoring and tracking struggle with image stabilization and camera obstruction, e.g., due to smoke, which produce significant errors in sensor measurements. Without addressing these errors properly, relying on such systems can be fatal.

Our work is closely relevant to the cooperative multi-robot target tracking literature. A comprehensive review of the taxonomy and recent approaches on multi-robot target detection and tracking is provided by Robin and Lacroix in [27]. An intensity function-based algorithm is presented in [28] to generate a control law for tracking dynamic targets with a group of UAVs. The problem of dynamic distributed task allocation in ground-robot teams for coordinated target tracking is studied in [29] through a bio-inspired approach. Mottaghi *et al.* [30] developed a particle filter-based approach to create a potential field to track a moving fire with multiple robots. Hausman *et al.* [31] developed a probabilistic localization and control method for a UAV team with fixed number of robots which seeks to minimize the expected future uncertainty of the target position. Both mutual information and EKF's covariance are investigated as measures of uncertainty.

A review of the recent methods on the automation methods for wildfire remote sensing application via UAVs is presented in [32], introducing and discussing three key metrics, i.e., situational awareness, decisional ability, and collaboration ability, in the recent relevant literature. Recent work by Sujit *et al.* [33] proposed a cooperative approach to detect and monitor multiple spots of fire (i.e. hotspots) using two groups of detector and service UAV agents. Afghah *et al.* [14], proposed a distributed leader-follower coalition formation model to cluster a set of UAVs into multiple coalitions that collectively cover a designated area. Kumar *et al.* [34] proposed a cooperative control algorithm to first cooperatively track the firefront shape for accurate situational awareness and then, autonomously fight the fire using fire suppressant fluid. Ghamry *et al.* [35] proposed a distributed algorithm with multiple stages such as search, confirmation and propagation monitoring for a team of UAVs to evenly distribute in a leader-follower manner to track an elliptical fire perimeter. Pham *et al.* [36] designed a fire heat-intensity based distributed control framework for a team of UAVs to be capable of closely monitoring a wildfire in open space in order to track its development. Harikumar *et al.* [37] proposes a search and dynamic formation control framework for a multi-UAV system to efficiently search for a dynamic target in an unknown environment. Pongpunwattana *et al.* [38] introduced a market-based cooperation planning system for a team of fixed-wing UAVs operating in a dynamic environment which accounts for uncertainty in future system states to compute tasks and path plans.

The majority of prior work on distributed control of UAVs, e.g. for MSN applications, focuses on maximizing the coverage of a particular area of interest [39, 40], such as an area of wildfire [41, 14]. Previous studies typically sought to maximize coverage (of a fire area) either through density-function based approaches [42, 43, 44, 5] or by maximizing the average pixel density across a terrain [45, 41]. In the latter case, UAV-based wildfire coverage studies tend to adopt a fire-intensity function for their coverage problem formulation [36, 39, 45, 5, 42, 43, 44] to model areas of fire according to an intensity model.

Many of the aforementioned approaches require an accurate fire-shape function [34] to work, or assume an enlarging elliptical perimeter for burning area to be monitored and tracked by moving UAVs [46, 33, 35, 37]. However, assuming a shape model for large-scale wildfires (when robot help is required) is not accurate [47]. Other approaches such as [36] use an "interestingness" function in order to enforce UAVs to look for areas of fire with predefined specifications. However, this function requires accurate, online measurements of heat intensity over the entire wildfire, which are often unavailable.

Previous image pixel density-based or intensity function-based approaches to distributed control of UAVs for field coverage have typically sought to maximize the fire coverage, as their sole objective [36, 14, 39, 40, 48, 49]. However, these approaches do not explicitly reason (i.e., through tracking and filtering) about a fire's state (i.e., position, velocity, scale and associated uncertainty), nor do they develop a predictive model for fire propagation by key parameters (e.g., fire-spread rate due to available fuel/vegetation and wind). Other approaches such as [15] and [16] utilized Kalman filter to track fire position on input image data, using frames centroids as model inputs. Nevertheless, unlike these estimation-based approaches, in chapter 5 we develop a predictive system with upper-bounds on the measurement-uncertainties that provides probabilistically-guaranteed performance.

Additionally, the fire-intensity based coverage approaches, such as [36] and [41], tend to generate a density-function reflecting the heat-intensity in different parts of a terrain. Density-functions and Voronoi Tessellation based coverage approaches are amongst the most popular methods for static or dynamic field coverage in which robots are distributed and assigned to different parts of a map to maximize coverage [39, 45, 5, 42, 43, 44]. Accordingly, fire-intensity based coverage approaches [36, 41] typically tend to model the location of a firefront as the "coolest" or "hottest" part of a fire visible via infrared sensors [36] according to the wildfire scenario, but this assumptions are not always accurate [50] for the applications of continuous monitoring and tracking. These methods utilize a fire heat intensity model to generate a time-varying density function and use artificial potential field to create driving force to control UAVs towards areas of minima (coolest part of fire) or maxima (hottest part of fire) [36]. Most of these methods are designed to cover as much of an area as possible with limited resources without systematically reasoning about the minimum required number robots for the task (our contributions in chapter 5) or prioritizing coverage areas based on task requirements or uncertainty of the environment (our contributions in chapter 4).

More closely along the line of our work in chapter 5, Bailon-Ruiz *et al.* [51] proposed a model-based planning algorithms to monitor a propagating wildfire using a fleet of UAVs. The approach tailors a variable neighborhood search to plan surveillance trajectories for a fleet of fixed-wing aircrafts according to a given fire propagation model and a given wind model. The fire state predictions are performed for hours into the future by using the integrated models and are updated through aircraft observations. Despite the similarities to our work in chapter 5, however, we did not consider the approach in [51] as a baseline for comparison, since the method assumes both a fire propagation model and a wind model to be given for making predictions and plans on the scale of hours into future and is designed for fixed-wing aircrafts with constrained motion dynamics. In our model-based approach in chapter 4 and chapter 5, we enable UAVs to actively infer fire propagation model parameters through environment observations and plan accordingly. In addition, the method in [51] directly plans trajectories for fixed-wing UAVs with constrained differential-drive dynamics which are not applicable to our presumed omni-directional UAVs in chapter 4 and chapter 5. The focus of our method in chapter 5, on the other hand, is to provide a performance-guaranteed plan and an upper-bound on the number of UAV agents needed to monitor the fire areas without losing the track quality.

2.1.2 Heterogeneous Multi-Robot Teams

Heterogeneity in robots' characteristics and roles are introduced to leverage the relative merits of different agents' capabilities [52, 9]. A group of heterogeneous robots that are collaborating on a number of co-dependent tasks to accomplish an overarching mission form a *composite robot team* [9, 53, 54]. Due to inherently different state- and action-spaces of agents in a composite team, communication, coordination, and consequently collaboration is not straightforward and requires proper considerations to model the interactions among heterogeneous agents. Additionally, such heterogeneity in a composite team implicitly entails a multi-faceted team objective such that several disjoint and sometimes competing objectives need to be successively and actively carried out to accomplish a mission.

Previous studies tackle various aspects of heterogeneous multi-robot teaming by considering coordination strategies [55], task allocation [56, 33], and path-planning and control [57, 8]. The problem of coordination and collaborative planning between perception and action agents has been of keen interest to the wireless communication research community, referring to the problem as *Wireless Sensor and Actor Networks* [53, 58]. While most of this prior work studies static environments (e.g., [55]), this assumption does not hold true in many environments of interest which are dynamic (e.g., including numerous, moving targets). Such dynamic environments have attributes of both a Partially Observable Markov Decision Process (POMDP) and a Restless Bandit Problem [59]. Unfortunately, traditional Reinforcement Learning (RL) formulations lack the scalability and adaptability with respect to domain shift in order to tackle real-world problems. Moreover, most of the proposed nonlearning-based approaches (e.g., mixed-integer linear/non-linear program) fail to handle the large-scale, dynamic, and stochastic nature of these problems.

Efficient planning and coordination of robots with different traits in a composite team while accounting for their collaborative behavior through specific capabilities and limitations is of significant importance [53, 52]. This coordination becomes more challenging when the dynamicity of the environment needs to be taken into account.

2.2 Learning End-to-End Multi-agent Coordination Policies

There has been large success in generating high-performing cooperative teams using MARL in challenging problems such as game playing [60] and robotics [61, 62]. In this section, we discuss the relevant prior work, including communication learning in MARL, application of Graph Neural Networks (GNN) in MARL and, heterogeneous multi-agent systems.

2.2.1 Enabling MARL with Communication

To learn cooperation protocols, prior MARL studies are commonly deployed under Decentralized Partially Observable Markov Decision Processes (Dec-POMDP), in which agents interact to maximize a shared discounted reward. Decentralized settings are typically preferred over central controllers since centralized approaches are costly, required extensive communication overhead and lack scalability and robustness to malicious attacks [63]. Since in a Dec-POMDP agents interact in a common environment to maximize a shared reward, the collaborative MARL objective (i.e., maximizing the discounted return accumulated by the team) is inherently satisfied. Nevertheless, such decentralized paradigms with the shared reward maximization objective introduce new challenges such as the credit assignment problem [64]. More recently, *fully-decentralized* (F-Dec) MARL was introduced [63, 65] to address the credit assignment problem caused by the shared-reward paradigm in conventional Dec-POMDPs [65, 64]. In an F-Dec MARL setting, agents can have varying reward functions corresponding to different tasks (e.g., in a multi-task RL where an agent solves multiple related MDP problems) which are only known to the corresponding agent and the collective goal is to maximize the globally averaged return over all agents. Nevertheless, under an F-Dec setting, agents seek to maximize their own reward, which does not necessarily imply the maximization of the team long-term return since agents do not inherently understand coordination.

Cooperative MARL studies can be subdivided into two main lines of research, (1) learning direct communication among agents to promote coordination [66, 67, 68, 69] and, (2) learning to coordinate without direct communication [70, 71, 10, 72]. Our work can be categorized under the former. Hierarchical approaches are also prevalent for learning coordination in MARL [5, 73, 74, 9]. We consider MARL problems in which the task in hand is of cooperative nature and agents can directly communicate, when possible.

Recently, the use of communication in MARL has been shown to enhance the collective performance of learning agents in cooperative MARL problems [75, 67, 68]. In recent years, several studies have been concerned with the problem of learning communication protocols and languages to use among agents. DIAL [66] and CommNet [68] displayed the capability to learn a discrete and continuous communication vectors, respectively. While DIAL considers the limited-bandwidth problem, neither of these approaches are readily applicable to composite teams or capable of performing attentional communication. TarMAC [67] achieves targeted communication through an attention mechanism which improves performance compared to prior work. Nevertheless, TarMAC requires high-bandwidth message passing channels and its architecture is reported to perform poorly in capturing the topology of interaction [76]. SchedNet [69] explicitly addresses the bandwidth-related concerns. However, in SchedNet agents learn how to schedule themselves for accessing the communication channel, rather than learning the communication protocols from scratch.

Graph Neural Networks (GNNs) are a class of deep neural networks that learn from unstructured data by representing objects as nodes and relations as edges and aggregating information from nearby nodes [77, 78, 79]. Prior work on MARL have sought to utilize a Graph Neural Networks (GNN) to model a decentralized communication structure among agents [80]. Deep Graph Network (DGN) [79] represents dynamic multi-agent interaction as a graph convolution to learn cooperative behaviors. This seminal work in MARL demonstrates that a graph-based representation substantially improves performance. In [81], an effective communication topology is proposed by using hierarchical GNNs to propagate messages among groups and agents. G2ANet [76] proposed a game abstraction method combining a hard and a soft-attention mechanism to dynamically learn interactions between agents. More recently, MAGIC [82] introduced a scalable, attentional communication model for learning a centralized scheduler. While these prior work have successfully modeled multi-agent interactions, they are not designed to address heterogeneous teams directly.

2.2.2 Modeling Heterogeneous Multi-Agent Systems via MARL

While MARL researchers have increasingly focused on developing computational models of team communication [75, 82], most of these prior frameworks fail to explicitly model the heterogeneity of *composite teams* and fail to explicitly quantify and reduce the team's communication overhead to support decentralized, bandwidth-limited teaming. Multi-agent communication is fundamental in composite teams, especially in the case where some agents are vision-limited. See [54, 9] for various examples of such composite teams. More accurately, we define a composite team as a group of heterogeneous agents that perform different tasks according to their respective capabilities while their tasks are co-dependent on accomplishing an overarching mission [83, 53, 9]. Agents in a composite team can inherently have different state, action, and observation spaces and yet, must still communicate essential information. Without a proper model for teaming, heterogeneous agents will not be able to reason about the heterogeneity in their team and share information accordingly to achieve team cognition. Therefore, communication may become unhelpful and deteriorate the MARL performance [84].

In [85], several types of heterogeneity induced by agents of different capabilities are discussed. As opposed to homogeneous teams, the diversity among agents in heterogeneous teams makes it challenging to hand-design intelligent communication protocols [85]. In [86], a control scheme is designed for a heterogeneous multi-agent system by modeling the interaction as a leader-follower system. While this approach is successfully applied to a UAV-UGV team, the control scheme is hand-designed and requires a fully connected communication structure. More recently, HMAGQ-Net [87] utilized GNNs and Deep Deterministic Q-network (DDQN) to facilitate coordination among heterogeneous agents (i.e., those with different state and action spaces).

2.2.3 Decision Rationalization and Mutual Information (MI) in MARL

In addition to communication, individuals in high-performing human teams also benefit from the theory of mind [88] and making strategic decisions by recursively reasoning about the actions (strategies) of other human members [89]. Such hierarchical rationalization alongside with communication facilitate meaningful cooperation in human teams [90]. In MARL, meaningful cooperation is an interactive policy through which the team objective is achieved. Similarly, collaborative Multi-agent Reinforcement Learning (MARL) relies on meaningful cooperation among interacting agents in a common environment [91]. Most of the prior works introduced in subsection 2.2.1 and subsection 2.2.2 on collaborative MARL are based on the maximum utility theory paradigm which assumes perfectly informed, rational agents [92]. Nevertheless, even under careful handcrafted or machine learned coordination policies, it is unrealistic and perhaps too strong to assume agents are perfectly rational in their decision-making [93, 94, 95, 96].

Recently, strong empirical evidence has shown that MI, defined as the information gain by an agent by observing another agent, is a statistic that correlates with the degree of collaboration between pairs of agents [97]. Researchers have also shown that maximizing MI among agents leads to maximizing the joint entropy of agents' decisions, which in turn, improves the overall performance in MARL [98]. As such, prior work has sought to increase MI by introducing auxiliary MI regularization terms to the objective function [98, 99]. These prior works adopt a centralized paradigm. Model of Other Agents (MOA) was proposed by [99] as a decentralized approach that seeks to locally push the MI lower-bound and promote collaboration among neighboring agents through predicting next-state actions of other agents. In all of the mentioned approaches, the amount of MI maximization objective that should be integrated into the overall policy objective is dictated through a β regularization parameter.

Among prior work seeking to enable k-level reasoning for MARL, [100] presented Probabilistic Recursive Reasoning (PR2), an opponent modeling approach to decentralized MARL in which agents create a variational estimate of their opponents' level $k-1$ actions and optimize a joint Q-function to learn cooperative policies without direct communication. [93] extended the PR2 for generalized recursive depth of reasoning. Neither of these works however, establish a link between k-level reasoning and MI for multi-agent coordination (our contribution in chapter 7).

2.2.4 Multi-Agent Learning from Expert Demonstration (MA-LfD)

Learning from Demonstration (LfD) explores techniques for learning a task policy from examples provided by a human teacher [101, 102]. Ho et al. [103] and Fu et al. [104] formulated the LfD problem under generative adversarial learning setting [105] to tackle the limitations in classic LfD frameworks such as BC [106], DAgger [107], and IRL [108]. Generative Adversarial IL (GAIL) [103] collects state-action pairs from executing the learned policy to shift the trajectories closer to the desired behavior. In GAIL, a *discriminator* model is trained to distinguish between state-action pairs provided by the expert and a deceiving *generator* model (i.e., the learned policy) that learns to imitate the expert. Standard RL algorithms are leveraged to optimize over the output of the discriminator (i.e., treated as a reward signal), encouraging the agent to match the expert-data in expectation, over full trajectories. Adversarial IRL (AIRL) [104] follows a setup similar to the GAIL but addresses the reward signal ambiguity in GAIL by leveraging a specific discriminator structure.

The literature for Multi-Agent LfD (MA-LfD) primarily aims to address the complexity of simultaneously training multiple agents. In [109], a coordinated IL approach is proposed which learns a latent coordination model along with the individual policies. In [110] the single-agent GAIL framework, described above, is extended for multi-agent scenarios along with a practical actor-critic method for multi-agent imitation. Yu et al. [111] extend the AIRL method to the multi-agent settings and propose a scalable framework. In [112] a scalable multi-agent LfD approach is proposed where a model-based heuristic method for automated swarm reorganization is leveraged to improve multi-agent task allocation problem. In [113] authors create an advising system to incorporate sub-optimal model-based heuristic policies to help improve MARL performance. More recently, Hoque et al. [114] proposed Fleet-DAgger , formalizing interactive fleet learning setting, in which multiple robots interactively query and learn from multiple human supervisors.

Nevertheless, applicability of these prior work in the collaborative multi-agent problems are considerably limited since none of these works explicitly address the inter-agent communication in complex domains where agents not only need to take task-related actions, but also need to communication and share information for coordination. Enabling inter-agent communication in these prior work requires the human expert to provide demonstrations for both environment actions and communication actions, which postulates an existing efficient communication strategy on a known message spaces. Additionally, none of these prior work consider a partially observable domain (common for realistic multi-robot systems) which necessitates the need for inter-agent communication and do not leverage real human-generated data for training to evaluate the approach against heterogeneity and variance in human data. These limitations can alleviate applicability of the mentioned works to multi-robot scenarios.

In our work in chapter 8, we address the limitations in prior work by relaxing the need for demonstrating a communication strategy by the expert. Using our method, a human expert can only teach the robot team how to accomplish a task collaboratively via demonstrations and the team will automatically learn a communication strategy suitable for the cooperation policy underlying the expert's demonstrations. The learned communication protocol will then help the robot team to deal with the partial observability, reasoning about action-decisions to best respond to teammates' policies, and alleviate the effects of environment non-stationarity. We also collect real human data and evaluate MixTURE's ability to cope with demonstration heterogeneity due to different expert styles and strategies.

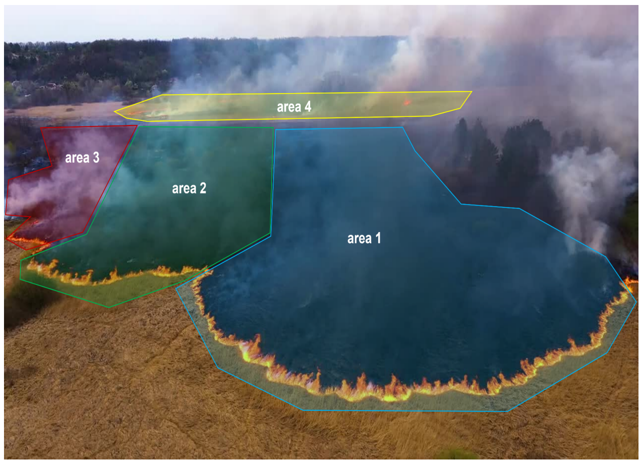
CHAPTER 3
PRELIMINARIES AND BACKGROUND

3.1 Motivating Applications

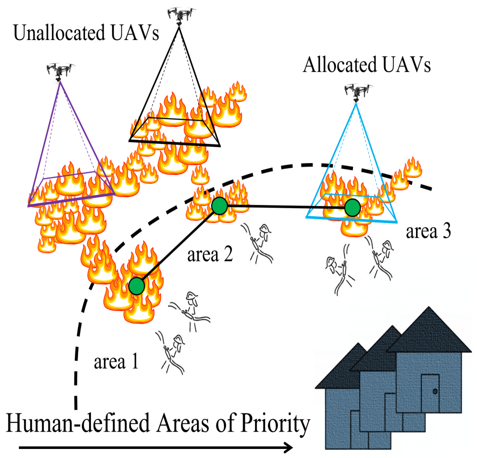
Throughout this thesis I predominantly motivate the multi-robot coordination methods in two applications: (1) aerial wildfire monitoring, where a group of homogeneous (similar in capabilities and tasks) Unmanned Aerial Vehicle (UAV)s are tasked with dynamically tracking and monitoring a propagating wildfire, and (2) wildfire fighting, where two groups of heterogeneous (different capabilities and objectives) robots, such as UAVs and Unmanned Ground Vehicles (UGVs), must collaborate to not only find, monitor, and track the fire, but also to extinguish them. Specifically for the latter case, I propose and create a novel multi-agent environment called FireCommander [54] details of which are presented in subsection 3.1.2. In the following I describe each of the two aforementioned applications and motivate my studies.

3.1.1 Aerial Wildfire Monitoring

I adopt the application of aerial wildfire monitoring as a running case-study and motivate my works in in chapter 4 and chapter 5 in this important safety-critical problem. In the application of wildfire fighting, human firefighters on the ground need online and dynamic observation of the firefront (i.e., the moving edge of fire) to anticipate a wildfire's unknown characteristics, such as size, scale, and propagation velocity, in order to plan their strategies accordingly. To support human firefighters, teams of UAVs can be deployed as Mobile Sensor Networks (MSN) to estimate the states of fire across thousands of acres and provide human firefighters with such information [7, 8, 15, 16, 14]. Nevertheless, coordinated control of UAV teams in this volatile setting provides particular challenges [17, 18, 19, 20, 21] and



(a) Clustering fire areas for monitoring based on propagation velocity and direction (courtesy of Motion Array; used with modifications).



(b) Separating fire areas for coverage and tracking based on areas of human activity and priority.

Figure 3.1: Figure 3.1a demonstrates an example of separating wildfire areas for coverage and tracking based on fire propagation velocity and direction. The firefront in each area is moving in a different direction and with a varying velocity and, therefore, at least one separate UAV is required to monitor and track the firefront in each area. In a similar perspective, Figure 3.1b shows an example of disjoint coverage areas that are separated according to areas of human (i.e., firefighters) activity and priority. In this case, the autonomous UAVs are teaming with humans to safely manage their human collaborators by providing for them *high-quality* information regarding their time-varying proximity to fire (i.e., areas of human operation). When we have access to additional UAV resources that may not be needed in the prioritized areas, we can deploy such UAVs (i.e., unallocated UAVs) to monitor the rest of the wildfire areas (not specified or prioritized).

an effective algorithm with the ability to prioritize coverage areas based on environment uncertainties and human firefighters' needs seems to be necessary (see section 4.1).

Fighting wildfires safely and effectively requires accurate online information on firefront location, size, shape, and propagation velocity [115, 116, 33, 15]. To provide firefighters with this realtime information, researchers have sought to utilize satellite feeds to estimate fire location information [117, 118, 46]. Unfortunately, the resolution of these images is too low for more than simple detection of a wildfire's existence [46]. Firefighters need frequent, high-quality images of the wildfire to make strategic plans [15, 36, 119]. Here, we define *high-quality information* as local, high-resolution images (or other sensory information) that are captured from a close by distance with respect to the areas prioritized by humans.

In large scale dynamic field coverage and tracking applications, such as aerial wildfire monitoring, UAVs need to be distributed effectively to cover the entire area of interest [5, 41]. In many cases, the region that need to be covered and tracked by the UAV team can be sub-divided into smaller areas according to different characteristics, priorities or needs. Figure 3.1a depicts an example of separating wildfire areas for coverage and tracking based on fire propagation velocity and direction. The firefront in each area is moving in a different direction and with a varying velocity and therefore, at least one separate UAV is required to monitor and track the firefront in each area. In a similar perspective, Figure 3.1b shows an example of disjoint coverage areas that are separated according to regions of human (i.e., firefighters) activity and priority. In this case, the autonomous UAVs are teaming with humans to safely manage their human collaborators by providing for them *high-quality* information regarding their time-varying proximity to fire (i.e., areas of human operation). Accordingly, when the UAV resources are limited, a multi-agent planning algorithm is required such that the real-time cooperative field coverage and tracking performance can be guaranteed with as few UAV agents as possible (see section 5.1).

3.1.2 Robotic Teams for Wildfire Fighting

In this application, at least two groups of heterogeneous (different capabilities and objectives) robots, such as UAVs and Unmanned Ground Vehicle (UGV)s, must collaborate to not only find, monitor, and track the fire, but also to extinguish them. We motivate the coordination of heterogeneous *composite* robot teams with perception and action agents in the problem of aerial wildfire fighting (see Figure Figure 3.2b). National firefighting departments such as the California Department of Forestry and Fire Protection (CAL FIRE) require online and accurate information about the wildfire states to plan for using large, limited, and high-cost aerial firefighting equipment such as Airtankers, Water Scoopers and Smokejumper aircraft or ground robots such as the Thermite RS1-T4 robotic firefighters [120]. These larger resource-rich aircrafts, equipped with fire retardant, can be considered as *Action* agents

(a) Firefighting Airtanker

(b) Composite UAV Teams for Aerial Wildfire Fighting

Figure 3.2: Figure Figure 3.2a depicts a firefighting Airtanker dropping fire retardant on the 2008 Tea Fire, Montecito, Calif. (Courtesy: National Public Radio, July 2012). Figure Figure 3.2b depicts a composite UAV team for aerial wildfire fighting. Smaller quadcopters (perception agents) sense the fire and gather information for larger, fixed-wing fire-extinguisher aircraft, i.e. action agents (Courtesy: Dronelife, Feb. 2016).

(or manipulators) which are capable of extinguishing large amounts of fire, however, are not suitable for estimating fire states due to high-velocity, high-altitude (Airtankers) or constrained ground vision (Thermite robots). To gather this required information on fire, however, other smaller, low-cost, and low-power devices such as multi-rotor quadcopters are more desirable due to their agility and cost-efficiency. These small information-gathering UAVs can perform as *Perception* agents (or sensing robots).

The challenges of coordinating a *Perception-Action* composite robot team is three-fold: (1) given an unknown dynamic wildfire environment, the robot team must balance the time spent on exploring new fire-areas (perception agents) and distinguishing the ones already found (action agents), (2) efficiently communicate the estimated fire states or distinguished firespots between perception and action agents to generated highly coordinated collaborative plans and policies, and (3) explicitly account for different agent characteristics (i.e., different state-, observation-, and action-spaces) and task objectives (exploring environment and finding areas of fire versus extinguishing firespots) in designing cooperative policies.

The FireCommander (FC) Multi-Agent Domain

To evaluate the performance of our methods, particularly the MARL algorithms proposed in chapter 5, chapter 6, and chapter 8, we design a new cooperative multi-agent environment with heterogeneous agents, called FireCommander [54, 121]. FireCommander can be categorized as a strategic game, in which a composite team of robots (i.e., UAVs and UGVs) must collaboratively find hidden areas of propagating wildfire and extinguish the fire in such areas as fast as possible. In FireCommander, two classes of *perception* and *action* agents must collaborate as a composite team to extinguish a propagating firespot. At each timestep, the firespot propagates to a new location according to the FARSITE [122] model, while the previous location is still on fire. All firespots are initially hidden to agents and need to be discovered before being extinguished. As such, *perception* agents are tasked to scan the environment to detect the firespots while *action* agents (no observation inputs) are required to move and extinguish a firespot that has been discovered by a *perception* agent before. Note that since firespots propagate, both *perception* and *action* agents need to continue to explore the map and collaborate until all firespots are extinguished. For more details regarding the environment and applications of such perception-action composite teams, please refer to the documentation [54] and its publicly available codebase on GitHub (available online: `https://github.com/CORE-Robotics-Lab/FireCommander`) [121].

3.2 FARSITE Fire Propagation Mathematical Model

Our model-based coordinated multi-UAV control and planning frameworks in chapter 4 and chapter 5 as well as our prevalent FireCommander domains [54, 121] relay on fire's approximate motion model. This model is leveraged in Kalman filter to infer its latent parameters through robots' observations. As such, we adopt the Fire Area Simulator (FARSITE) wildfire propagation mathematical model [122]. The FARSITE model is now widely used by federal and state land management agencies in the United States.

In this thesis, we leverage the *simplified* FARSITE model [36, 5, 41] to predict firefront propagation and use dynamic observations to infer latent parameters of the model. We note that although our model-based frameworks in chapter 4 and chapter 5 have access to the fire propagation model, this environment model does not necessarily need to be the FARSITE model and can be replaced with any dynamic target motion model. For instance, the FARSITE wildfire propagation model used in chapter 4 and chapter 5 or the FireCommander domain [54, 121] can be replaced with any other parameterized model, such as the correctable fire simulation model introduced in [123] which calculates the fire spread based on Rothermel's original surface fire model [124]. In the following we present the details of the *simplified* FARSITE model as utilized in prior work [36, 41].

Considering q_t^i as the location of i-th firespot on firefront at time t and $\dot{q}_t^i$ as a firefront's growth rate at location q_t^i (i.e., fire propagation velocity), the wildfire propagation dynamics can be written as in Equation 3.1, where δt is the time-step and $\dot{q}_t^i = \frac{d}{dt}\left(q_t^i\right)$ is a function of fire spread rate (R_t, i.e., fuel and vegetation coefficient), wind speed (U_t), and wind azimuth (θ_t), which are available to our system through weather forecasting equipment.

$$q_t^i = q_{t-1}^i + \dot{q}_{t-1}^i \delta t \tag{3.1}$$

In Equation 3.1, by ignoring the superscript i and without losing generality, the firefronts growth rate, $\dot{q}_t$, can be estimated for each propagating spot on firefront by Equation 3.2 - Equation 3.3, where $\dot{q}_t^x$ and $\dot{q}_t^x$ are first-order firefront dynamics for X and Y axes [122, 5].

$$\dot{q}_t^x = C(R_t, U_t)\sin(\theta_t) \tag{3.2}$$

$$\dot{q}_t^y = C(R_t, U_t)\cos(\theta_t) \tag{3.3}$$

In above equations, $C(R_t, U_t)$ is a firespot's velocity which is a function of the fuel/vegetation coefficient, R_t, and mid-flame wind speed, U_t. We use equations from Finney [122] to calculate C as in Equation 3.4, in which $LB(U_t) = 0.936e^{0.256U_t} + 0.461e^{-0.154U_t} - 0.397$ and

$GB(U_t) = LB(U_t)^2 - 1.$

$$C(R_t, U_t) = R_t \left(1 - \frac{LB(U_t)}{LB(U_t) + \sqrt{GB(U_t)}}\right) \tag{3.4}$$

Furthermore, due to the fuel exhaustion, we model the intensity decay of a fire spot during its ignition time δt_q, as a dynamic exponential decay rate λ over time, as $I^q_{t+\delta t} = I^q_t \left(e^{-\lambda \frac{\delta t_q}{R_t}}\right)$. In this equation, I^q_t is the heat intensity of firespot q at time t and is calculated according to the intensity model proposed by [125], in which $I^q_t = 259.833 \left(\frac{h^q_t}{\cos(\alpha^q_t)}\right)^{2.174}$. In this equation, h^q_t and α^q_t are flame height (meters) and tilt angle (degrees) with respect to vertical horizon line for each firespot and the intensity I^q_t is measured in kilo-watts per meter.

Finally, when a firefighting UAV or UGV drops the extinguisher fluid over an area of fire, we cut the fire intensities of the respective fire spots according to a predefined extinguisher fluid coefficient. A fire point is pruned from the fire-map if its intensity falls below a threshold value, leaving a burnt spot on the terrain map which cannot catch fire anymore.

3.3 Markov Decision Processes and Reinforcement Learning

In its basic form, a Markov Decision Process (MDP) can be represented via a 6-tuple, $\langle \mathscr{S}, \mathscr{A}, \mathscr{R}, \mathscr{T}, \gamma, \rho 0 \rangle$. $\mathscr{S}$ and $\mathscr{A}$ are the state- and action-space, respectively. $\mathscr{R} : \mathscr{S} \to \mathbb{R}$ is the reward function. At each timestep, t, the agent takes an action, $a_t \in \mathscr{A}$ in state $s_t \in \mathscr{S}$ which leads to a state transition according to the state transition probability density function, $\mathscr{T}(s_{t+1}|s_t, a_t) : \mathscr{S} \times \mathscr{A} \to \mathscr{S}$. The agent is rewarded $\mathscr{R}(s_t, a_t) \in \mathbb{R}$ after this transition. $\gamma \in [0, 1)$ is the temporal discount factor and $\rho 0 : \mathscr{S} \to \mathbb{R}$ denotes the initial state probability. The goal in an MDP is to find a good policy for the decision maker. A policy $\pi : \mathscr{S} \times \mathscr{A} \to \mathbb{R}$ is a mapping from states to probabilities over actions.

RL is a machine learning training method based on rewarding desired or penalizing undesired behaviors. A reinforcement learning agent can perceive its environment and interpret the feedback received, take actions, and eventually learn through trial and error [126].

An RL agent can generate a trajectory $\tau = \langle s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots \rangle$ by executing a policy, π, within the environment. The objective for RL is to solve this sequential decision-making process by finding an optimal policy, π^*, such that the expected discounted return of the policy is maximized. This optimization problem can be formulated as in Equation 3.5.

$$\pi^* = \arg\max_{\pi} J(\pi) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \mathscr{R}(s_t, a_t) \right] \tag{3.5}$$

3.3.1 Policy Gradient Methods

Policy Gradient (PG) methods are an approach to RL that utilize function approximation, in which each agent j has a policy, $\pi_{\varphi}^{j}(a|s)$, parameterized by φ, that specifies which action, a, to take in each state, s, to maximize the expected future discounted reward. PG methods apply gradient ascent to the *actor's* (i.e., policy) parameters, φ, based on a gradient estimate of the expected return in Equation 3.5. By the policy gradient theorem [126], the expected reward maximization, $J(\varphi)$, is maximized via Equation 3.6, where a_t^j and o_t^j are the action and observation of agent j, respectively.

$$\nabla_{\varphi} J(\varphi) = \mathbb{E}_{\pi_{\varphi}^{j}} \left[\nabla_{\varphi} \log \pi_{\varphi}^{j}(a_t^j | o_t^j) \hat{A}_t(o_t^j, a_t^j) \right] \tag{3.6}$$

The *advantage* function, $\hat{A}_t(o_t^j, a_t^j) = Q(o_t^j, a_t^j) - V^{\phi}(o_t^j)$, measures how much better the action is relative to the default action of the current policy [127], where $Q(o_t^j, a_t^j)$ is the action value function approximated by the total discounted rewards, and $V^{\phi}(o_t^j)$ is the state-value function, approximated according to a critic network parametrized by ϕ.

3.4 Multi-Agent Reinforcement Learning

Prior work typically models the learning process in MARL as a Multi-Agent Partially Observable Markov Decision Process (MA-POMDP) [128]. This can be represented by a 9-tuple $\langle \mathscr{N}, \mathscr{S}, \mathscr{A}, \Omega, \mathscr{O}, r, \mathscr{T}, \gamma \rangle$. $\mathscr{N}$ is the total number of agents, $\mathscr{S} = \{\times_i \mathscr{S}^i\}_{i=1}^{\mathscr{N}}$

is a joint set of agent state-spaces. Note that this joint state- space increases linearly as the environment size becomes larger and exponentially as the number of agents in the environment increases. For each agent j we can represent the state vector at time t as $\mathscr{S}_t^j = \left[s_t^{j_1}, s_t^{j_2}, \cdots, s_t^{j_m}\right]$, where m is the state vector dimension. $\mathscr{A} = \{\times_i \mathscr{A}^i\}_{i=1}^{\mathscr{N}}$, is the joint set of action-spaces. $\Omega = \{\times_i \Omega^i\}_{i=1}^{\mathscr{N}}$ is similarly defined as the joint set of observation-spaces. $\gamma \in [0,1)$ is the temporal discount factor for each unit of time and $\mathscr{T}$ is the state transition probability density function: $\mathscr{T} : \mathscr{S} \times \mathscr{A} \rightarrow \mathscr{S}$. Note that in a homogeneous setting (i.e., similar agents) the contents of state-, observation- and actions-spaces are the same for all agents while in a heterogeneous multi-agent setting [13], the class (i.e., type) of an agent determines the content of its state-, action- and observation-spaces, such as the number of or the type of variables in these spaces.

At each timestep, t, each agent, j, can receive a partial observation $o_t^j \in \Omega$ according to an observation function $\{\mathscr{O}\} : o_t^j \sim \mathscr{O}(\cdot|\bar{s})$. If the environment observation is not available for agents of class i, agents in the respective class will not receive any input from the environment (e.g., lack of sensory inputs). Regardless of receiving an observation or not, at each time, t, each agent, j, takes an action, a_t^j, forming a joint action vector $\bar{a} = \left(a_t^1, a_t^2, \cdots, a_t^j\right)$. This step leads to changing the joint states to $\bar{s}' \in \mathscr{S}$ according to the state transition probability density function $\mathscr{T}\left(\bar{s}_{t+1}|\bar{s}_t, \bar{a}_t\right)$. Next, depending on the next joint-state, $\bar{s}_{t+1}$, they receive an immediate reward, $r(\bar{s}_t, \bar{a}_t) \in \mathbb{R}$, shared by all agents. We note that Such shared reward, encourages collaboration and teaming behaviour among agents [69]. Our objective is to learn an optimal policy, $\pi_j^*(s_t^j) : \mathscr{S} \rightarrow \mathscr{A}$ that maximizes the total expected, discounted reward accumulated by agents over an infinite horizon, or as formally presented in Equation 3.7.

$$\pi^*(\bar{s}) = \underset{\pi(\bar{s}) \in \Pi}{\arg\max}\, \mathbb{E}_{\pi(\bar{s})} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | \pi(\bar{s}) \right] \tag{3.7}$$

3.5 Learning from Demonstration

Learning from Demonstration (LfD) explores techniques for learning a task policy from examples provided by a human teacher [101, 102]. Behavioral Cloning (BC) is one of the simplest approaches to learn from demonstrations in which an agent is directly trained on an expert-provided dataset to maximize the likelihood of the demonstrated actions given the corresponding environment states [129]. However, when training data is limited or the environment is highly stochastic, BC suffers from poor performance due to compounding errors resulting in covariate shift [130][1].

Dataset Aggregation (DAgger) [107] attempts to address the limitations of BC through interactive data collection. In other words, DAgger proceeds by collecting a dataset at each iteration under the current policy and trains the next policy under the aggregate of *all* collected trajectories. Intuitively, through this process DAgger attempts to build up the set of inputs that the learned policy is likely to encounter during its execution [107]. Nevertheless, DAgger makes the assumption that a simulator of the environment as well as an expert are always accessible so that they can interactively collect new state-action pairs. These assumptions are strong and not always feasible.

Generative Adversarial Imitation Learning (GAIL) [103] addresses the aforementioned issues by collecting state-action pairs from executing the learned policy to shift the trajectories closer to the desired behavior. GAIL casts the optimization in a generative adversarial framework, learning a *discriminator* model, D_θ, to distinguish between state-action pairs provided by the expert and a deceiving *generator* model (i.e., the learned policy), π, that learns to imitate the expert. Standard RL algorithms are leveraged to optimized over the output of the discriminator (i.e., treated as a reward signal), encouraging the agent to match the expert-data in expectation, over full trajectories. However, once the learned policy closely matches the demonstrated behavior, the discriminator's output in GAIL does not

[1]In states not seen in the demonstration dataset agents tend to have poor performance and since actions taken in the current state affect future states, the errors tends to compound over the course of a trajectory.

necessarily encode useful information.

The *imitation learning* approaches described above aim to learn a policy which effectively imitates the demonstrated expert behavior. In contrast, a paradigm known as Inverse Reinforcement Learning (IRL) [108] instead attempts to extract a reward function which explains the demonstrated behavior [131]. IRL [108] is another solutions to learn from expert provided demonstrations which considers an MDP sans reward function. The goal of the IRL is to infer and recover a reward function $R(s,a)$ given a set of demonstration trajectories $U = \{\tau_1, \tau_2, \cdots, \tau_N\}$, where N is the number of demonstrations. A typical assumption for IRL is that demonstrated trajectories are optimal, or at least near-optimal.

Adversarial Inverse Reinforcement Learning (AIRL) [104] solves the IRL problem with a generative-adversarial setup similar to the GAIL framework described above. In AIRL, the discriminator, D_θ, is trained via a Binary Cross Entropy (BCE) loss and predicts whether the state transition (s_t, s_{t+1}), belongs to the demonstrator dataset or is generated by the generator model, $\pi_\varphi(a|s)$. The generator, π_φ, is trained to maximize the pseudo-reward given by the discriminator [132]. AIRL addresses the reward signal issue in GAIL by leveraging a specific discriminator structure which enables recovering a meaningful reward function even after convergence. The discriminator in AIRL is given by the Equation 3.8 where $\hat{f}_\theta(s,a)$ is the inferred reward function parameterized by θ, and $\pi_\varphi(a|s)$ is the learned policy parametrized by φ.

$$D_\theta = \frac{\exp\left(\hat{f}_\theta(s,a)\right)}{\exp\left(\hat{f}_\theta(s,a)\right) + \pi_\varphi(a|s)} \tag{3.8}$$

The inferred reward function in Equation 3.8, $\hat{R} = \hat{f}_\theta(s,a)$, is updated using gradient descent via the BCE loss function shown in Equation 3.9. Minimizing the BCE loss in Equation 3.9 allows the discriminator, D_θ, to distinguish expert trajectories from generator policy rollouts. The policy, $\pi_\varphi(a|s)$, is trained to imitate the expert by maximizing the pseudo-reward function given by $\hat{R} = \hat{f}_\theta(s,a)$.

$$L_D = -\mathbb{E}_{\tau \sim D,(s,a)\sim\tau}\left[\log D(s,a)\right] - \mathbb{E}_{\tau\sim\pi,(s,a)\sim\tau}\Big[\log\big(1 - D(s,a)\big)\Big] \tag{3.9}$$

We use the introduced concepts, in the following chapters to develop solutions and design architectures to model collaborative multi-robot teams under various environment constraints and task objectives. Particularly, we leverage MARL and RL in chapter 5-chapter 7 for learning collaborative policies directly from data experienced through environment interactions. I later utilize LfD in chapter 8 to tackle the weaknesses of MARL and teach robots to collaborate directly via human-expert generated data.

CHAPTER 4
MODEL-BASED NODE-LEVEL CONTROL FOR MULTI-ROBOT COORDINATION

Collaborative field coverage is a widely applied instance of a classic multi-robot coordination problem. In these problems, a robot team is deployed as a MSN [133]. MSNs are instances of wireless sensor networks [133] with robots as their mobile nodes. MSNs are versatile and can cope with rapid topology changes. Due to their agility and hovering capabilities, UAVs are good candidates to be deployed as MSNs [5].

To enable a highly efficient and intelligent team behavior, in [5] (i.e., section 4.1), we design a low-level control strategy toward a human-centered robot coordination. Such a system is desired in a variety of applications; as an example, deploying a fleet of UAVs to actively monitor a propagating wildfire in support of human firefighters on the ground. In [5], we propose a decentralized control framework that leverages a model-predictive mechanism to coordinate a UAV team for tracking the moving firefronts while simultaneously enabling the firefighters to receive online information regarding their time-varying proximity to fire.

An issue that we faced in [5] was that a connected communication graph was required at all times for the control architecture to work properly. The team's communication network, however, may be disconnected at times and its links may have varying strengths due to environment constraints. In [6] (i.e., section 4.2), we tackle this problem by designing a model-reference multi-agent adaptive controller that achieves team convergence even for a network of robots with a *disconnected* communication graph or even non-communicative robots. We derive an adaptive control law for a leader-follower networked system that provably converges to mimic any desired network structure even though the real communication topology remains unknown to the robots.

4.1 Coordinated Control of UAVs for Human-Centered Active Sensing of Wildfires

Fighting wildfires is a precarious task, imperiling the lives of engaging firefighters and those who reside in the fire's path. Firefighters need online and dynamic observation of the firefront to anticipate a wildfire's unknown characteristics, such as size, scale, and propagation velocity, and to plan accordingly (see subsection 3.1.1 for a detailed discussion). In this paper, we propose a distributed control framework to coordinate a team of UAVs for a human-centered active sensing of wildfires. We develop a dual-criterion objective function in Equation 4.3 based on Kalman uncertainty residual propagation and weighted multi-agent consensus protocol, which enables the UAVs to actively infer the wildfire dynamics and parameters, track and monitor the fire transition, and safely manage human firefighters on the ground using acquired information. We empirically evaluate our approach relative to prior work as well as on physical robots in a mock fire-monitoring exercise [5].

4.1.1 Introduction and Motivation

Fighting wildfires requires accurate online information regarding firefront location, scale, shape, and propagation velocity [115, 33]. Firefighters may lose their lives as a consequence of inaccurately anticipating information either due to inherent stochasticity in fire behavior or low-quality information provided, such as low-resolution satellite images [46]. Firefighters need frequent, high-quality images to monitor the fire propagation and plan accordingly. As such, Multi-robots teams capable of fast scheduling and task allocation [134, 135] with analytical temporal upper-bounds [7, 8, 136] are of particular interest for this applications. Due to recent advances in aerial robotic technology, UAVs have been proposed as a solution to overcoming the challenges of needing real-time information in fighting fires [137].

In [33], a cooperative approach is proposed to detect local fire areas in a wildfire using two groups of detector and service UAV agents. In [25], utility of visual and infrared cameras on heterogeneous UAVs with hovering capabilities is investigated to monitor the

Figure 4.1: Firefighters trying to control a back burn as the Carr fire spreads toward Douglas City. Courtesy of the Los Angeles Times (May 2019).

evolution of the firefront shape. In [35], a leader-follower-based distributed framework is proposed for a team of UAVs to evenly distribute and track an elliptical fire perimeter. In [36], a heat-intensity-based distributed control framework is designed for a team of UAVs to be capable of closely monitoring a wildfire in open space. More recently, both model-based (i.e., Kalman estimation) and model-free (e.g., learning [138]) methods have been used for cooperative prediction and tracking of the firefront shape [139, 140, 34]. Additionally, other learning-based approaches, such as RL, have also been applied to this problem to enable collaborative monitoring of wildfires [119, 141].

There is a clear absence of human-centric approaches (e.g., [142, 143, 144]) in the literature for UAV teams for active sensing of wildfire and firefront monitoring. This is mainly because a majority of previous studies are solely focused on autonomous fire detection and surveilling a large burning area by drones rather than focusing on local human-defined areas of priority (i.e., areas of firefighter activity) and serving human firefighters by taking into account their safety. In this study, we seek a control strategy toward a smarter, human-centered robot coordination, through better perception and accurate local situational awareness. We overcome key limitations in prior work by developing an algorithmic framework to provide a model-predictive mechanism that enables firefighters to receive online, high-quality information regarding their time-varying proximity to a fire.

Contributions

In our approach, we explicitly estimate the latent fire propagation dynamics and parameters via an adaptive extended Kalman filter (AEKF) predictor and the simplified FARSITE wildfire propagation model [122] to account for firefighter's safety and provide them with online information regarding propagating firefronts. This model allows us to develop straightforward distributed control adapted from vehicle routing literature [145] to enable track-based fire coverage. Moreover, a mathematical observation model through which UAV sensors observe fire is derived to map from state space to observation space. The calculated models are then used in combination to derive a dual-criteria objective function in order to control a fleet of UAVs. The proposed dual-criteria objective is an ad hoc, well-suited function to the wildfire monitoring task, which minimizes environment's uncertainty on local, human-centered areas (first criteria) and maximizes coverage through ensemble-level formation control of the robot network (second criteria).

We empirically evaluate our approach against simulated wildfires alongside contemporary approaches for UAV coverage [36] as well as against a reinforcement learning baseline, demonstrating a promising utility of our method. Our coordinated controller is capable of effectively reducing the cumulative uncertainty residual of the fire environment in firefront coverage to support human-robot teaming. We also assess the feasibility of our method through implementation on physical robots in a mock firefighting scenario.

4.1.2 Preliminaries: Adaptive Extended Kalman Filter (AEKF)

We utilize an AEKF to leverage the mathematical fire propagation model and the observation model of a flying drone with respect to a dynamic object on the ground to actively sense the fire-spots, infer wildfire dynamics and parameters, and propagate all sources of measurement uncertainty. In a conventional Extended Kalman Filter (EKF), process and observation noise covariances Q_t and Γ_t are often chosen as constant matrices based on the state-transition model and sensor accuracy and do not receive updates. However, this selection process is highly sensitive to user experience and can be extremely inaccurate. We leverage AEKF [146], which introduces *innovation* and *residual*-based updates for process and observation noise covariances, as shown in Equation 4.1-Equation 4.2, where α is a forgetting factor and d_t is the measurement innovation and is defined as the difference between the actual measurement and its predicted value. Moreover, K_t is the Kalman gain and H_t is the observation Jacobian matrix.

$$Q_t = \alpha Q_{t-1} + (1-\alpha)\left(K_t d_t d_t^T K_t^T\right) \tag{4.1}$$

$$\Gamma_t = \alpha \Gamma_{t-1} + (1-\alpha)\left(\tilde{y}_t \tilde{y}_t^T + H_t P_{t|t-1} H_t^T\right) \tag{4.2}$$

These adaptive updates remove the assumption of constant covariances Q_t and Γ_t and enable even more accurate predictions over time as the Kalman filter leverages its observations to improve the predicted covariance matrix $P_{t|t-1}$ [146].

4.1.3 Problem Statement and Algorithmic Overview

The focus of our study includes two important aspects of wildfire monitoring: (1) providing high-quality information on firefront status while accounting for physical and methodological errors and (2) human-centered coverage and tracking of wildfire to account for firefighter safety. Accordingly, we define high-quality information as high-resolution and online images of areas prioritized by humans. We take advantage of the estimated uncertainty of the environment to achieve these objectives. Through a unified error propagation system, not only can the physical and methodological uncertainties be leveraged to manage the human teams and account for their safety, they can also be used to manage the UAV team, both in node-level and ensemble-level dynamics. An AEKF is a proper candidate for the uncertainty propagation system here since it can accumulate physical and methodological errors and generate a cumulative error map through the calculated probability distribution and predicted covariance matrix.

Accordingly, UAVs initially calculate two uncertainty maps: (1) a firefront uncertainty map (subsubsection 4.1.4) and (2) a human uncertainty map (subsubsection 4.1.4). The latter is generated through a bimodal distribution of human locations as received by GPS devices while the first map is created by the AEKF's online inference of firefront locations and error propagation (subsection 4.1.5). Through the combination of these two error-maps, we obtain our first node-level controller (uncertainty-based controller, subsubsection 4.1.4). We also incorporate an ensemble-level (formation) controller to encourage the UAV team to maintain a formation consensus for maximizing the coverage (subsubsection 4.1.6). The two controllers coordinate to generate a virtual position for each UAV which is then fed to a path planning controller to generate the force required to move the UAV to the determined position based on UAVs flight dynamics (subsection 4.1.6).

In algorithm 1 we depict an overview of the proposed human-centered coordinated control procedure for monitoring wildfires. Upon receiving a request, UAVs travel to the human-defined areas of interest (i.e., rendezvous point, line 1 in algorithm 1). On arrival, UAVs sense the firefront by extrapolating fire-spots q_t and generate a general uncertainty map $\mathscr{U}_t^{tot}$ by fusing AEKF error propagation and areas of human activity using GPS data (line 3-5 in algorithm 1). Afterwards, a combination of an uncertainty-based optimization and a graph-based weighted consensus protocol forms our new dual-criteria objective function $\mathscr{H}$, shown in Equation 4.3 for a set of N UAVs (line 6 in algorithm 1). In Equation 4.3 p_t^d represents UAV positions and Q is the entire fire-map. This objective function is then embedded as our coordinated control system to move UAVs to highly uncertain areas on the generated error map to minimize the associated error while encouraging drones to maintain a distributed formation to increase the team efficiency in field coverage (lines 7-8 in algorithm 1). To generate the required control inputs for UAVs to move, we calculate the negative derivative of the objective function with respect to the location of drones at time t. Meanwhile, AEKF is also used to infer the firefront characteristics, such as spatial distribution $\hat{q}_t$, propagation velocity $\dot{q}_t$, and direction, in order to calculate an individualized temporal safety index (SI) (Equation 4.37) for firefighters.

$$\min \mathscr{H} = \underset{q_t^i, p_t^d}{\arg\max} \left(\int_{i\in Q} \mathscr{U}_t^{tot}\left(q_t^i\right) dq - \int_{d\in N} \mathscr{E}_{dj}\left(\left\|p_t^d - p_t^{j/d}\right\|\right) dp \right) \tag{4.3}$$

Algorithm 1: Stages of the proposed human-centered coordinated control for a team of UAVs.

```
   input :Obtain the rendezvous area $p_r$, fire-map $Q_t$, human GPS data $p_t^h$, UAV positions $p_t^d$, and fire
          propagation model $\mathscr{M}$
1  Move to rendezvous area: $p_t^d \leftarrow \texttt{Move}(p_r, p_{t-1}^d)$
2  while MissionDuration do
3      Generate firefront uncertainty map: $\mathscr{U}_t^f, q_t \leftarrow \texttt{Sense}(Q_t)$
4      Generate human uncertainty map: $\mathscr{U}_t^h \leftarrow \texttt{GPS}(q_h)$
5      Combine uncertainty maps: $\mathscr{U}_t^{tot} = \mathscr{U}_t^f + \mathscr{U}_t^h$
6      Minimize the dual-criteria objective function:
           $\min \mathscr{H} = \underset{q_t^i, p_t^d}{\arg\max} \left( \int_{i\in Q} \mathscr{U}_t^{tot}\left(q_t^i\right) dq - \int_{d\in N} \mathscr{E}_{dj}\left(\left\|p_t^d - p_t^{j/d}\right\|\right) dp \right)$
7      Calculate the overall control inputs and determine new virtual positions for UAVs:
           $p_{t+\delta t}^n = p_t^v - \left(u_d^{ucc} - u_d^{fcc}\right)\delta t$
8      Move to the new desired position: $p_{t+\delta t}^d \leftarrow \texttt{Move}(p_{t+\delta t}^n, p_t^d)$
9  end
10 def Move $(p_g^d, p_t^d)$:   // $p_g^d$ is the goal position for drone $d$
11     $u_{d,t} = \sum_i u_{d,t}^{att_i}(p_g^d, p_t^d) + u_{d,t}^{rep_i}(p_g^d, p_t^d), \ \forall i \in F_t$
12     $p_{t+\delta t}^d = p_t^d + u_{d,t}\delta t$
13 def Sense $(Q_t)$:   // $\mathscr{O}_t$ is the UAV observation model
14     $\hat{q}_t = \arg\max_{q_t} \rho\left(q_{t|t-1}, p_{t-1}, \mathscr{M}_{t-1}, \mathscr{O}_{t-1}\right)$
```

4.1.4 Method

The following sections are dedicated to discussing and formulating the two modules of the dual-criteria objective function in Equation 4.3, as well as elaborating on the uncertainty map generation process.

Criteria 1: Uncertainty-based Controller

We design our coverage and tracking controller to minimize the uncertainty of the firefront locations over time, while focusing on the areas of human operation. To this end, we generate two uncertainty maps for (1) the propagating fire locations $\mathscr{U}_t^f$ and (2) the areas of human activity $\mathscr{U}_t^h$. Eventually, we fuse these error maps together by linearly summing up the respective estimated uncertainty values of each point to obtain the general uncertainty map $\mathscr{U}_t^{tot}$ at time t. The uncertainty-based controller's objective is to minimize the overall uncertainty residual in $\mathscr{U}_t^{tot}$. We present the details of calculating $\mathscr{U}_t^f$ and $\mathscr{U}_t^h$ in the following sections. Figure 4.2 demonstrates the formation of the uncertainty map and the foundation of our human-centered controller.

Firefront Uncertainty Map

We leverage AEKF to estimate a probability distribution of the fire-spot locations and compute a measurement covariance for each point through linear error propagation techniques. Considering q_{t-1} as the location of firefronts at current time and p_{t-1}^d as the UAV coordinates, a firefront location $\hat{q}_t$ one step forward in time is desired, given the current firefront distribution (q_{t-1}), fire propagation model with current parameters ($\mathscr{M}_{t-1}$), and UAV observation model of the field ($\mathscr{O}_{t-1}$) as in Equation 4.4

$$\hat{q}_t = \arg\max_{q_t} \rho \left(q_{t-1}, p_{t-1}, \mathscr{M}_{t-1}, \mathscr{O}_{t-1}, q_t\right) \tag{4.4}$$

In AEKF, the uncertainty of the firefront locations over time is measured as the state covariance $P_{t|t}$ at time t. It has been shown previously (see [147]) that minimizing the state covariance corresponds to maximizing the covariance residual S_t in Equation 4.5 where $P_{t|t-1}$ is the predicted covariance, F_t and H_t are process and observation model Jacobians, and Q_t and Γ_t are the corresponding noise covariances and can be calculated as $P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + Q_t$.

$$S_t = H_t P_{t|t-1} H_t^T + \Gamma_t \tag{4.5}$$

According to Equation 4.5, by setting $p_{t|t-1}$ to identity, we see that a maximally informative position for drones is the one that minimizes the $H_t H_t^T$, or in other words, the closest possible position where dynamic observations change rapidly [147]. As such, we generate an uncertainty map which is reflective of the wildfire dynamics where a measurement residual can be calculated for each point q_t by summing up the estimated covariance residual matrix S_t and set our objective to maximize S_t. Accordingly, we derive our new objective function $\mathscr{U}_t^f$ as in Equation 4.6, where $\mathrm{Tr}(.)$ represents the trace operation, and fov_t^d is the field of view (FOV) of drone d at time t.

$$\begin{aligned}\min \mathscr{U}_t^f &= \underset{q_t^i \in fov_t^d}{\arg\max} \left(\mathrm{Tr}\left(S_t\right)\right) = \underset{q_t^i \in fov_t^d}{\arg\max} \left(\mathrm{Tr}\left(H_t P_{t|t-1} H_t^T + \Gamma_t\right)\right) \\ &= \underset{q_t^i \in fov_t^d}{\arg\max} \left(\mathrm{Tr}\left(H_t\left(F_t P_{t-1|t-1} F_t^T + Q_t\right) H_t^T + \Gamma_t\right)\right)\end{aligned} \tag{4.6}$$

Human Uncertainty Map

While hovering around the highly uncertain areas to provide firefighters with online information regarding the firefront, UAVs are required to focus on their human collaborators on the ground and take their safety into account by putting additional concentration on the areas of human operation. Accordingly, UAVs receive human positions p_t^h (i.e., through GPS devices) at time t as planar coordinates $\mu_{t,h}^x$ and $\mu_{t,h}^y$ and generate a bimodal Gaussian distribution for each human h to account for both error in GPS information as well as the

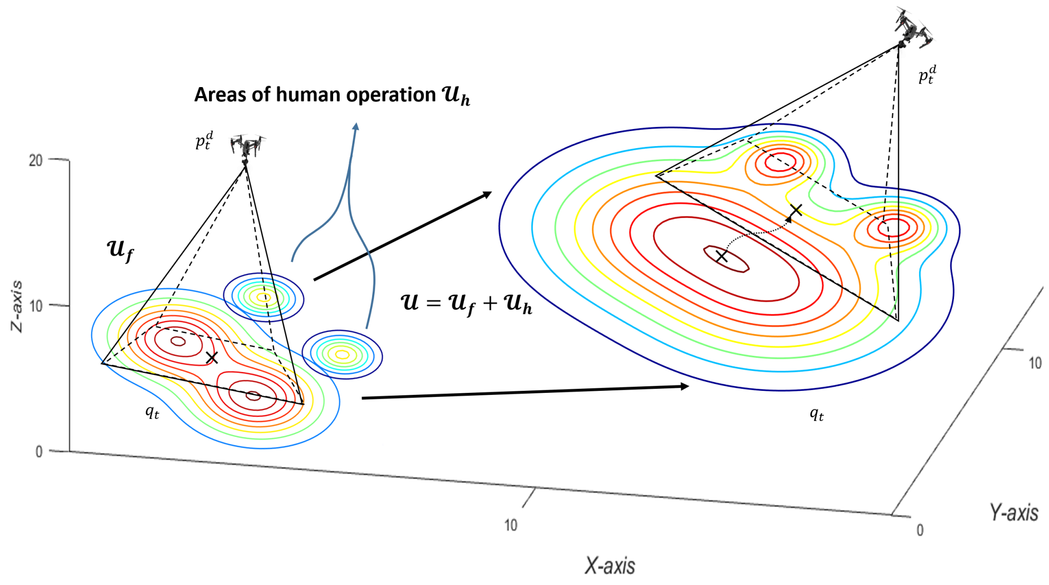

Areas of human operation $\mathcal{U}_h$
p_t^d
$\mathcal{U}_f$
$\mathcal{U} = \mathcal{U}_f + \mathcal{U}_h$
q_t
Z-axis
X-axis
Y-axis
20
10
0

Figure 4.2: Fusing AEKF measurement residual and human GPS data to generate an uncertainty map. UAVs focus on areas of human activity while monitoring fire propagation.

mobility of the humans. By assuming independence, a joint PDF can be calculated as our human safety objective as in Equation 4.7 where v_j represents the points in a safe circular vicinity of human h with radius r_s and P_s is a predefined safety threshold. P_{ih} is a Cumulative Distribution Function (CDF) with respect to each human location and all approaching fires.

$$\mathscr{U}_t^h = \underset{q_t \in v_h}{\arg\max} \prod_{i \in v_h} P_{ih} \left\{ q_t^i - p_t^h \geq r_s \right\} \geq P_s \tag{4.7}$$

To calculate the P_{ih}, we leverage the estimated fire-spot locations q_t alongside inferred fire parameters (i.e., $\hat{R}_t$, $\hat{U}_t$, and $\hat{\theta}_t$) by AEKF to calculate a glsCDF for each human at location p_t^h and all approaching firefronts q_t^i. We then integrate the resulting glsCDF to be greater than the safe-distance. A Probability P_{ih} is then calculated for each individual as in Equation 4.8 where $\mu_{q_t^i}$ and $\sigma_{t,i}^2$ are calculated by AEKF. Equation 4.7 is leveraged later in subsubsection 4.1.7 to compute an individualized safety index for each firefighter.

$$P_{ih} = \int_{\tau=r_s}^{\infty} \mathscr{N}\left(\mu_{q_t^i} - p_t^h, \sigma_{t,i}^2\right) d\tau = 1 - \text{CDF}\left(r_s | \mu_{q_t^i} - p_t^h, \sigma_{t,i}^2\right) \tag{4.8}$$

Criteria 2: Weighted Multi-agent Consensus Protocol

To ensure that the combination of the above objective functions results in local actions leading up to appropriate global performance, we enforce an extra control term in such a way that the UAVs also act on other easily measurable information, such as the relative displacements to neighboring drones. This is specifically important to disperse UAVs, while preserving the connectedness of the network, from converging to an extreme minima (i.e., a highly uncertain point). Accordingly, we leverage the weighted consensus protocol [148] as in Equation 4.9 for a set of N UAVs with the objective of minimizing the total displacement error $\mathscr{E}_{ij}$ while preserving a distance of at least δ between all UAVs.

$$\min \mathscr{E} = \underset{p_t}{\arg\min}\, \mathscr{E}_{ij}\left(\left\|p_t^i - p_t^j\right\|\right) = \underset{p_t}{\arg\min} \sum_{i=1}^{N} \sum_{j \in V_i} \frac{1}{2(\Delta - \delta)} \left(\frac{\left\|p_t^i - p_t^j\right\| - \delta}{\Delta - \left\|p_t^i - p_t^j\right\|} \right)^2 \tag{4.9}$$

In Equation 4.9, $j \in V_i$ represents $j-$th UAV within the communication range Δ of UAV i. As such, a negative force will be generated to move UAVs apart from or closer to each other if they are getting closer than δ or farther than Δ (i.e., the UAV network becomes disconnected). Note that δ should be set high enough so that the UAV team can spread effectively.

4.1.5 Online Inference of Wildfire Dynamics

The joint probability density function in Equation 4.4 is calculated through AEKF estimator. Using the aforementioned notations, the AEKF state transition and observation equations can now be stated as in Equation 4.10 and Equation 4.11 where p_{t-1} is the UAV location.

$$q_t = f_{t-1}(q_{t-1}, p_{t-1}, R_{t-1}, U_{t-1}, \theta_{t-1}) + \omega_t \tag{4.10}$$

$$\hat{q}_t = h_t(q_t, p_{t-1}) + \nu_t \tag{4.11}$$

We reform the state transition equation in Equation 4.10 to account for all state variables in $\Theta_t = \left[q_t^x, q_t^y, p_t^x, p_t^y, p_t^z, R_t, U_t, \theta_t\right]^T$ as in Equation 4.12 where the process noise ω_t.

$$\begin{bmatrix} \Theta_t \end{bmatrix}_{8\times 1} = \begin{bmatrix} \left.\frac{\partial f}{\partial \Theta_i}\right|_{\hat{\Theta}_{t-1|t-1}} \end{bmatrix}_{8\times 8} \begin{bmatrix} \Theta_{t-1} \end{bmatrix}_{8\times 1} + \omega_t,\ \forall i \in \Theta \tag{4.12}$$

Therefore, we form the state transition Jacobian matrices F_t as in Equation 4.13, including partial derivatives of wildfire propagation dynamics in Equation 3.1 with respect to all state variables.

$$\left.\frac{\partial f}{\partial \Theta_i}\right|_{\hat{\Phi}_{t'}} = \begin{array}{c} \\ q_t^x \\ q_t^y \\ p_t^{(3)} \\ R_t \\ U_t \\ \theta_t \end{array} \begin{array}{c} \begin{array}{cccccc} q_{t'}^x & q_{t'}^y & p_{t'}^{(3)} & R_{t'} & U_{t'} & \theta_{t'} \end{array} \\ \begin{pmatrix} 1 & 0 & 0^{(3)} & \frac{\partial q_t^x}{\partial R_{t'}} & \frac{\partial q_t^x}{\partial U_{t'}} & \frac{\partial q_t^x}{\partial \theta_{t'}} \\ 0 & 1 & 0^{(3)} & \frac{\partial q_t^y}{\partial R_{t'}} & \frac{\partial q_t^y}{\partial U_{t'}} & \frac{\partial q_t^y}{\partial \theta_{t'}} \\ 0^{(3)} & 0^{(3)} & 0^{(3)} & 0^{(3)} & 0^{(3)} & 0^{(3)} \\ 0 & 0 & 0^{(3)} & 1 & 0 & 0 \\ 0 & 0 & 0^{(3)} & 0 & 1 & 0 \\ 0 & 0 & 0^{(3)} & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{4.13}$$

In Equation 4.13 $t' = t-1$ and superscript (3) represent number of column and row repetitions for all $\left[p_t^x, p_t^y, p_t^z\right]$. We note that the parameters R_t, U_t, and U_t are not necessarily dynamic with time, and it is fairly reasonable to consider these physical parameters as constants for short periods of time. However, in the case of analyzing the system for longer durations, temporal dynamics may apply [149], specifically due to changes in wind speed and velocity. Exact estimation of temporal dynamics related to these parameters are out of the scope of the current study, since we assume locality in time and space according to FARSITE [122]. The partial derivatives of q_t^x and q_t^y with respect to parameters R_{t-1}, U_{t-1}, and θ_{t-1} are computed by applying the chain-rule and using Equation 3.2 - Equation 3.3 as shown in Equation 4.14 - Equation 4.16, where $\mathscr{L}(\theta)$ equals $\sin\theta$ and $\cos\theta$ for X and Y

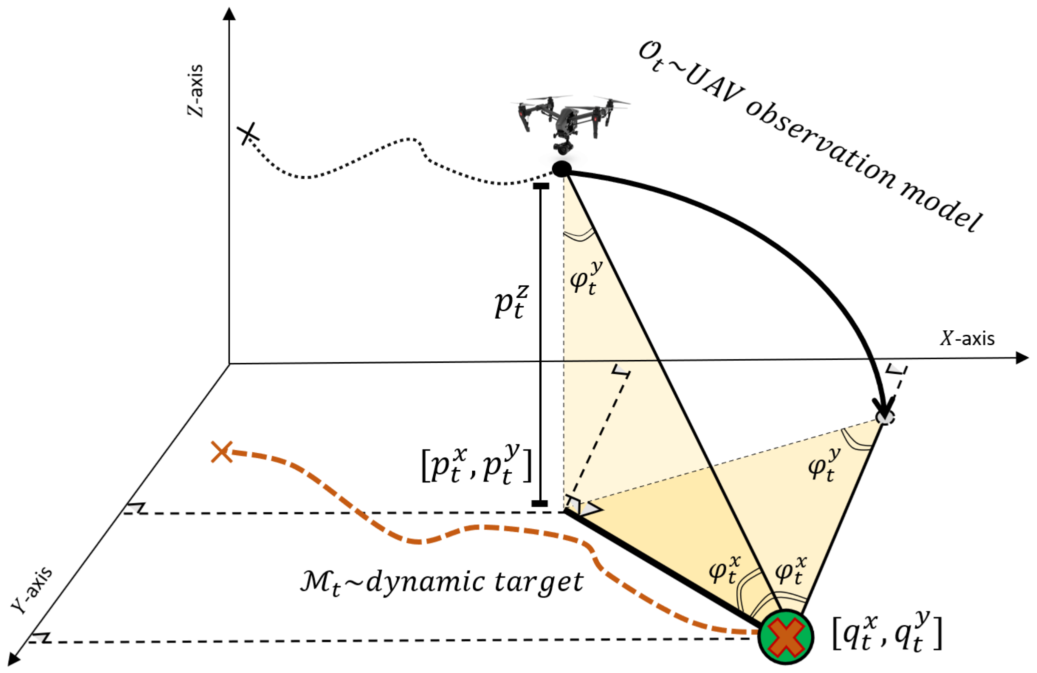


Figure 4.3: The UAV observation model.

axis, respectively.

$$\frac{\partial q_t}{\partial \theta_{t-1}} = C(R_t, U_t)\frac{\partial \mathscr{L}(\theta)}{\partial \theta}\delta t \tag{4.14}$$

$$\frac{\partial q_t}{\partial R_{t-1}} = \left(1 - \frac{LB(U_t)}{LB(U_t) + \sqrt{GB(U_t)}}\right)\mathscr{L}(\theta)\delta t \tag{4.15}$$

$$\frac{\partial q_t}{\partial U_{t-1}} = \frac{R_{t'}\left(LB(U_{t'})\frac{\partial GB(U_{t'})}{\partial U_{t'}} - GB(U_{t'})\frac{\partial LB(U_{t'})}{\partial U_{t'}}\right)}{\left(LB(U_{t'}) + \sqrt{GB(U_{t'})}\right)^2}\mathscr{L}(\theta)\delta t \tag{4.16}$$

Next, we derive the observation model through which UAVs perceive dynamic fire-spots, according to Figure 4.3. The observation mapping in Equation 4.11 is reformed into Equation 4.17 where $\Phi_t = \left[\varphi_t^x, \varphi_t^y, \hat{R}_t, \hat{U}_t, \hat{\theta}_t\right]^T$ is a mapping vector through which the estimated parameters are translated into a unified, observed *angle*-parameter vector $\hat{\Phi}_t$.

$$\begin{bmatrix}\hat{\Phi}_t\end{bmatrix}_{5\times 1} = \begin{bmatrix}\left.\frac{\partial h}{\partial \Theta_i}\right|_{\Phi_{t|t}}\end{bmatrix}_{5\times 8}\begin{bmatrix}\Phi_t\end{bmatrix}_{8\times 1} + v_t,\ \forall i \in \Theta \tag{4.17}$$

According to Figure 4.3, the angle parameters are calculated as $\varphi_t^x = \tan^{-1}\left(\frac{p_t^z}{\|q_t - p_t\|}\right)$ and $\varphi_t^y = \tan^{-1}\left(\frac{\|q_t - p_t\|}{p_t^z}\right)$ for X and Y axes respectively, where $q_t = [q_t^x, q_t^y]$ and $p_t = [p_t^x, p_t^y]$. Then, the observation Jacobian matrix H_t is calculated as in Equation 4.18

$$\left.\frac{\partial h}{\partial \Theta_i}\right|_{\hat{\Theta}_{t|t'}} = \begin{array}{c|cccccccc} & q_t^x & q_t^y & p_t^x & p_t^y & p_t^z & R_t & U_t & \theta_t \\ \hline \varphi_t^x & \frac{\partial \varphi_t^x}{\partial q_t^x} & \frac{\partial \varphi_t^x}{\partial q_t^y} & \frac{\partial \varphi_t^x}{\partial p_t^x} & \frac{\partial \varphi_t^x}{\partial p_t^y} & \frac{\partial \varphi_t^x}{\partial p_t^z} & 0 & 0 & 0 \\ \varphi_t^y & \frac{\partial \varphi_t^y}{\partial q_t^x} & \frac{\partial \varphi_t^y}{\partial q_t^y} & \frac{\partial \varphi_t^y}{\partial p_t^x} & \frac{\partial \varphi_t^y}{\partial p_t^y} & \frac{\partial \varphi_t^y}{\partial p_t^z} & 0 & 0 & 0 \\ \hat{R}_t & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \hat{U}_t & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hat{\theta}_t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \tag{4.18}$$

Using the aforementioned angle parameter equations, the partial derivatives for X-axis are derived as in Equation 4.19 - Equation 4.21.

$$\nabla_{q_t}\varphi_t^x = \frac{1}{1+\left(\frac{p_t^z}{\|q_t-p_t\|}\right)^2}\left(\frac{-p_t^z(q_t-p_t)}{\|q_t-p_t\|^3}\right) = \left[\frac{\partial \varphi_t^x}{\partial q_t^x}, \frac{\partial \varphi_t^x}{\partial q_t^y}\right] \tag{4.19}$$

$$\nabla_{p_t}\varphi_t^x = \frac{1}{1+\left(\frac{p_t^z}{\|q_t-p_t\|}\right)^2}\left(\frac{p_t^z(q_t-p_t)}{\|q_t-p_t\|^3}\right) = \left[\frac{\partial \varphi_t^x}{\partial p_t^x}, \frac{\partial \varphi_t^x}{\partial p_t^y}\right] \tag{4.20}$$

$$\frac{\partial \varphi_t^x}{\partial p_t^z} = \frac{1}{1+\left(\frac{p_t^z}{\|q_t-p_t\|}\right)^2}\left(\frac{1}{\|q_t-p_t\|}\right) \tag{4.21}$$

and for Y-axis derivatives, we can derive as in Equation 4.22 - Equation 4.24.

$$\nabla_{q_t}\varphi_t^y = \frac{1}{1+\left(\frac{\|q_t-p_t\|}{p_t^z}\right)^2}\left(\frac{(q_t-p_t)}{p_t^z\|q_t-p_t\|}\right) = \left[\frac{\partial\varphi_t^y}{\partial q_t^x}, \frac{\partial\varphi_t^y}{\partial q_t^y}\right] \tag{4.22}$$

$$\nabla_{p_t}\varphi_t^y = \frac{1}{1+\left(\frac{\|q_t-p_t\|}{p_t^z}\right)^2}\left(\frac{-(q_t-p_t)}{p_t^z\|q_t-p_t\|}\right) = \left[\frac{\partial\varphi_t^y}{\partial p_t^x}, \frac{\partial\varphi_t^y}{\partial p_t^y}\right] \tag{4.23}$$

$$\frac{\partial\varphi_t^y}{\partial p_t^z} = \frac{1}{1+\left(\frac{\|q_t-p_t\|}{p_t^z}\right)^2}\left(\frac{-\|q_t-p_t\|}{(p_t^z)^2}\right) \tag{4.24}$$

The process noise ω_t in Equation 4.12 accounts for both stochasticity in fire behavior and wildfire propagation model inaccuracy. Moreover, the observation noise v_t is responsible to account for the estimation errors associated with both q_t and p_t^d which affect UAVs' ability to extrapolate where a fire is on the ground. Both ω_t and v_t are modeled as a zero-mean white Gaussian noise with covariances Q_t and Γ_t, respectively. Note that errors in X, Y, and Z axes coordinates of a drone are loosely correlated, and thus, we also incorporate non-diagonal elements in noise covariance matrices when initializing them. Q_t and Γ_t then receive adaptive updates according to AEKF framework, in Equation 4.1 and Equation 4.2.

4.1.6 Controller Design

Figure 4.4 represents our node-level controller architecture for each UAV d with neighboring UAVs $p_t^{i/d}$. Our controller consists of three components: (1) an Uncertainty Control Component (UCC), (2) a Formation Controller Component (FCC), and (3) a Path Planning Controller (PPC). The first controller performs exploitation to minimize the overall uncertainty in the map (i.e., firefront locations or human areas of activity) while the second controller is designed to manage the general formation of the UAV swarm in order to maximize exploration as well as coverage. The third controller component moves UAVs to any desired next position. Assuming $u_d = \dot{p}_d$ to be the quadcopter UAV dynamics, we develop each of our control components in the following sections.

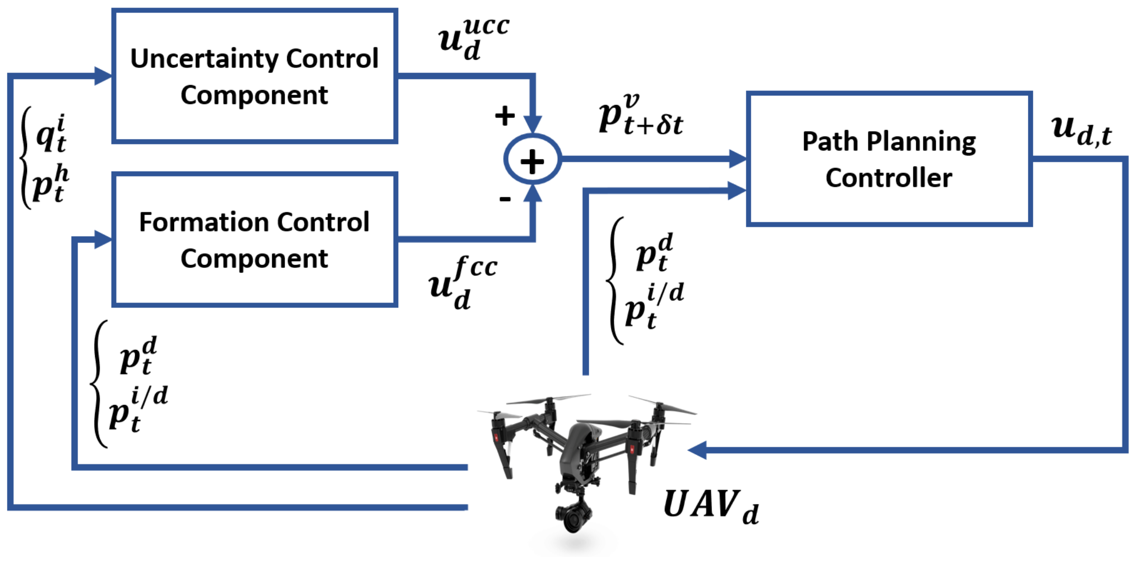


Figure 4.4: Node-level controller architecture of each drone d.

Uncertainty Controller Component (UCC)

The first controller works based on the theory of artificial potential field [150] where each UAV is distributedly controlled by a negative gradient of the generated total uncertainty map $\mathscr{U}_t^{tot}$ from objective functions in Equation 4.6 and Equation 4.9, with respect to its position $p_t^d = \left[p_t^x, p_t^y, p_t^z\right]^T$ as follows in Equation 4.25 where κ_1 is the proportional gain parameter.

$$u_d^{ucc} = -\kappa_1 \frac{\partial \mathscr{U}_t^{tot}}{\partial p_d} \tag{4.25}$$

To derive the gradients with respect to the UAV coordinates, we first need to analytically derive the uncertainty objective function (Equation 4.6). To do so, we insert the values of process and observation Jacobian matrices (i.e., F_t and H_t) and the process and observation noise covariances (i.e., Q_t and Γ_t) and calculate the trace of the final matrix (see subsection 4.1.5). Eventually, after the simplifications, the UCC objective function equation can be derived as in Equation 4.26 where the gradient terms can be calculated using introduced angle-parameters.

$$\min \mathscr{U} = \underset{q_t^i \in fov_t^d}{\arg\max} \left(\beta_1 \left(\frac{\partial \varphi_t^x}{\partial q_t^x} \right)^2 + \beta_2 \left(\frac{\partial \varphi_t^y}{\partial q_t^y} \right)^2 + \beta_3 \left(\frac{\partial \varphi_t^x}{\partial p_t^x} \right)^2 \right.$$
$$\left. + \beta_4 \left(\frac{\partial \varphi_t^y}{\partial p_t^y} \right)^2 + \beta_5 \left(\left(\frac{\partial \varphi_t^x}{\partial p_t^z} \right)^2 + \left(\frac{\partial \varphi_t^y}{\partial p_t^z} \right)^2 \right) \right) \tag{4.26}$$

In Equation 4.26 β_i are covariance constants and are equal to $\beta_1 = \left(P_{11} + \sigma_{q^x}^2\right)$, $\beta_2 = \left(P_{22} + \sigma_{q^y}^2\right)$, $\beta_3 = \sigma_{p^x}^2$, $\beta_4 = \sigma_{p^y}^2$ and $\beta_5 = \sigma_{p^z}^2$. Accordingly, the final gradients in Equation 4.25 with respect to UAV pose can be calculated as in Equation 4.27- Equation 4.29

$$\frac{\partial \mathscr{U}}{\partial p_d^x} = \beta_1 \frac{\partial \left(\frac{\partial \varphi_t^x}{\partial q_t^x} \right)^2}{\partial p_d^x} + \beta_3 \frac{\partial \left(\frac{\partial \varphi_t^x}{\partial p_d^x} \right)^2}{\partial p_d^x} + \beta_5 \frac{\partial \left(\frac{\partial \varphi_t^x}{\partial p_d^z} \right)^2}{\partial p_d^x} \tag{4.27}$$

$$\frac{\partial \mathscr{U}}{\partial p_d^y} = \beta_2 \frac{\partial \left(\frac{\partial \varphi_t^y}{\partial q_t^y} \right)^2}{\partial p_d^y} + \beta_4 \frac{\partial \left(\frac{\partial \varphi_t^y}{\partial p_d^y} \right)^2}{\partial p_d^x} + \beta_5 \frac{\partial \left(\frac{\partial \varphi_t^y}{\partial p_d^z} \right)^2}{\partial p_d^y} \tag{4.28}$$

$$\frac{\partial \mathscr{U}}{\partial p_d^z} = \beta_1 \frac{\partial \left(\frac{\partial \varphi_t^x}{\partial q_t^x} \right)^2}{\partial p_d^z} + \beta_2 \frac{\partial \left(\frac{\partial \varphi_t^y}{\partial q_t^y} \right)^2}{\partial p_d^z} + \beta_3 \frac{\partial \left(\frac{\partial \varphi_t^x}{\partial p_d^x} \right)^2}{\partial p_d^z}$$
$$+ \beta_4 \frac{\partial \left(\frac{\partial \varphi_t^y}{\partial p_d^y} \right)^2}{\partial p_d^z} + \beta_5 \left(\frac{\partial \left(\frac{\partial \varphi_t^x}{\partial p_d^z} \right)^2}{\partial p_d^z} + \frac{\partial \left(\frac{\partial \varphi_t^y}{\partial p_d^z} \right)^2}{\partial p_d^z} \right) \tag{4.29}$$

Eventually, the control input to UAV d from UCC module is shown in Equation 4.30

$$u_{d,t}^{ucc} = \left[\kappa_x \frac{\partial \mathscr{U}_t^{tot}}{\partial p_d^x}, \kappa_y \frac{\partial \mathscr{U}_t^{tot}}{\partial p_d^y}, \kappa_z \frac{\partial \mathscr{U}_t^{tot}}{\partial p_d^z} \right] \tag{4.30}$$

We note that there is no need to explicitly calculate the gradients of human uncertainty map with respect to UAV positions separately, since we linearly sum up the values (non-negative) of the two maps (see Figure 4.2).

Formation Controller Component (FCC)

Similar to the UCC, our formation controller component (FCC) attempts to minimize the consensus displacement error by using a gradient descent flow of the weighted consensus protocol in Equation 4.9, with respect to UAV pose, as represented in Equation 4.31.

$$u_d^{fcc} = -\kappa_2 \frac{\partial \mathscr{E}}{\partial p_d} = -\sum_{(j,d)\in E} \frac{\left(1 - \frac{\delta}{\left\| p_t^d - p_t^{d/j} \right\|}\right)\left(p_t^d - p_t^{d/j}\right)}{\left(\Delta - \left\| p_t^d - p_t^{d/j} \right\|\right)^3} \tag{4.31}$$

Similar logistics as in Equation 4.30 can be derived here for the three axes of coordinate. Accordingly, the combination of control inputs generated by UCC and FCC modules are leveraged according to our dual-criteria objective function, introduced in Equation 4.3, in order to produce a new desired location p_t^v for each UAV to move to. As such, a UAV's new virtual position will be updated and fed to PPC as shown in Equation 4.32.

$$p_{t+\delta t}^v = p_t^v - \left(u_d^{ucc} - u_d^{fcc}\right)\delta t \tag{4.32}$$

Path Planning Controller (PPC)

The PPC module generates either an attractive force toward a desired pose or a repulsive force avoiding an undesirable one. Desired poses include the initial rendezvous point where coverage and tracking wildfire begins and the new virtual position $p_{t+\delta t}^v$ generated through our dual-criteria objective function as in Equation 4.32. Undesirable poses include ones that are too close to another UAV or too high/low of an altitude. Leveraging an artificial potential field, we address these problems by generating attractive and repulsive forces using a quadratic function of distances from desired or to undesired points. The attractive control force applied to each UAV p_t^d to any goal points p_t^g at time t can be calculated as noted in

Equation 4.33 where κ_g is the proportional gain.

$$\mathscr{D}_d^{att} = \frac{1}{2}\kappa_g \left\| p_g^d - p_t^d \right\|^2 \text{ and } u_d^{att} = -\nabla \mathscr{D}_d^{att} = \kappa_g \left(p_g^d - p_t^d \right) \tag{4.33}$$

Using the same notation, the repulsive control force generated to avoid any point p_t^g can be defined as in Equation 4.34 where γ is the distance between current position and the undesirable position p_t^g and $\zeta = 1$ only if $\left\| p_t^g - p_t^d \right\| < \gamma$.

$$\mathscr{D}_d^{rep} = \begin{cases} \frac{1}{2}\kappa_g \left(\frac{1}{\left\| p_g^d - p_t^d \right\|} - \frac{1}{\gamma} \right)^2 & if \quad \left\| p_g^d - p_t^d \right\| < \gamma \\ 0 & \text{otherwise.} \end{cases} \tag{4.34}$$

$$u_d^{rep} = -\zeta \nabla \mathscr{D}_d^{rep} = -\zeta \kappa_g \left(\frac{1}{\left\| p_g^d - p_t^d \right\|} - \frac{1}{\gamma} \right) \frac{1}{\left\| p_g^d - p_t^d \right\|^3} \left(p_g^d - p_t^d \right) \tag{4.35}$$

Eventually, the general control law in order to generate the required force to move UAVs to their new locations can be formed as in Equation 4.36 where F_t is the set of all generated attractive and repulsive forces at time t. Thus, the final position of UAV d gets updated through $p_{t+\delta t}^d = p_t^d + u_d \delta t$.

$$u_{d,t} = \sum_i u_{d,t}^{att_i} + u_{d,t}^{rep_i}, \ \ \forall i \in F_t \tag{4.36}$$

4.1.7 Empirical Evaluation

Safety Index to Secure Human Firefighters

As a corollary of our algorithm, we calculate an individualized safety index (SI) as a temporal quantity for human firefighters on the ground by leveraging the estimated wildfire dynamics and parameters and report this quantity to firefighters for situational awareness. Now, SI as a measure of time is defined as in Equation 4.37 for human firefighters by taking into account

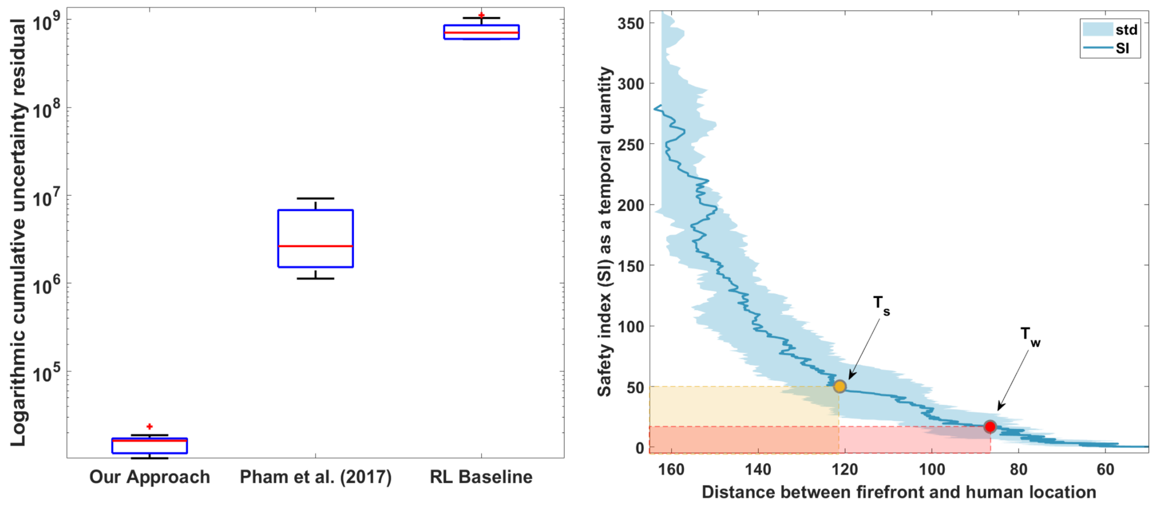


Figure 4.5: This figure depicts a quantitative comparison of our coordinated controller with prior work (left-side) and human safety index (SI) variations as a temporal quantity, with respect to distance between an approaching firefront and a human firefighter (right-side).

the velocity of the approaching firefront.

$$SI_t^h = \prod_{i \in v_h} P_{ih} \left(\dot{q}_t^i \frac{p_t^h - q_t^i}{\left\| p_t^h - q_t^i \right\|} \right)^{-1} \tag{4.37}$$

In this equation, P_{ih} is the CDF (from Equation 4.8), v_h is the vicinity of human h, and $\dot{q}_t^i$ is the estimated fire spread velocity of fire-spot i toward this vicinity. The ratio is to account for the direction of the firefront and equals to 1 if the firefront is directly approaching the coordinates where the human is located. Accordingly, we assume three different ranges for SI to be announced at each time, namely (1) safe if $SI_t^h \geq T_s$, (2) warning if $T_w \leq SI_t^h < T_s$, and (3) danger if $SI_t^h < T_w$. Parameters T_s and T_w are predefined temporal-bounds for safety and warning situations, respectively. We leave the safety and warning thresholds T_s and T_w to be pre-defined by humans, as these variables are subjective to the firefighting scenario (e.g., a burning hospital versus forest fire) and are dependent on situational severity. The right-side figure in Figure 4.5 depicts the variations (i.e. mean$\pm$std) of SI with respect to distance between an approaching firefront with 10 points and a human firefighter over 100 trials of simulation. For this case, a single UAV was placed over the fire area, inferring the fire-spot locations and parameters.

Results

We evaluate the efficiency of our controller in simulation and against two benchmarks: (1) a state-of-the-art, model-based, distributed control [36] and (2) a deep reinforcement learning (RL) baseline. The first benchmark [36] is a fire heat-intensity-based distributed control framework for wildfire coverage which incorporates FARSITE and a model for fire heat-intensity measure in order to maximize the area-pixel density of the UAV's fire observations. Furthermore, we train an RL policy network to control UAVs to reduce the uncertainty residual as measured by AEKF. The network consists of four convolutional layers followed by three fully connected layers with ReLU activations. The image of the fire area is an input while a direction for UAVs is an output of the network. We define the reward at each step as the negative sum of uncertainty residual across the entire map, encouraging the agent to minimize uncertainty over time.

In our simulations, we initialize the fire-map with 20 randomly placed ignition points in [50 100] range and within a 500-by-500 terrain where the fire model parameters R_t, U_t, and θ_t were chosen similar to [36], for comparison. A total of five drones were initialized around [50 300] coordinates with initial altitude set to zero. UAV camera half-angles were set to $[\frac{\pi}{4}, \frac{\pi}{6}]$. The inter-distance δ and communication range Δ in our weighted consensus protocol were set to 50 and 500, respectively. The maximum and minimum altitudes were chosen to be 15 and 45, respectively. Figure 4.6 depicts the simulation results of eight sample time-steps between $t = 20$ (top-left) and $t = 300$ (bottom-right) as detailed above, representing drone FOVs projected on the ground.

The left-side figure in Figure 4.5 shows a comparison for a team of UAVs controlled by our method, the distributed control proposed by [36], and the RL baseline. We ran the simulations for 100 time-steps for all three methods for a total of 10 trials where for each trial, a cumulative uncertainty was calculated by the AEKF for fire points not covered by any

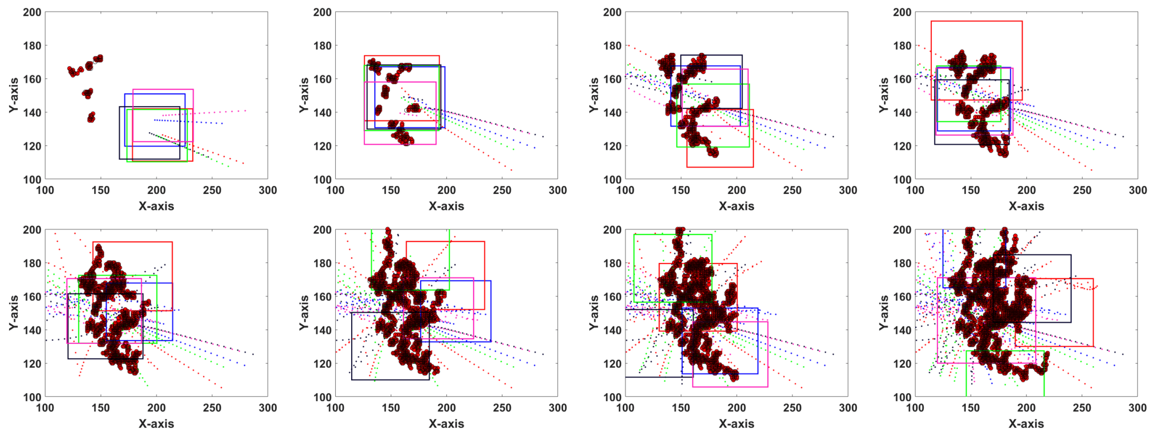


Figure 4.6: Simulation results of eight sample time-steps between $t = 20$ (top-left) and $t = 300$ (bottom-right) for distributed coverage, representing drone FOVs projected on the ground. Dot rays show the FOV centroids.

drones at each step. While the RL baseline failed to learn during 800 episodes of training, our approach shows significant improvements by reducing the cumulative uncertainty residual by more than 10^2x and 10^5x times.

We also evaluate the feasibility of our controller on physical robots. The physical experiments with actual robots were performed in the Robotarium, a remotely accessible swarm robotics research platform [151]. We tested the coverage performance of our controller using five robots and similar fire environment as above. Figure 4.7 represents example demonstrations of our experiment. Results of the experiment is demonstrated in the supplementary video, which can also be found at `https://youtu.be/j3YdIO5u_fE`.

4.1.8 Conclusion

We combined a node-level control criteria and an ensemble-level control criteria to introduce a novel coordinated control algorithm for human-centered active sensing of wildfires, providing high-quality, online information to human firefighters on the ground. In our approach, we take advantage of AEKF's error propagation capability to generate an uncertainty map, incorporating uncertainties about firefront dynamics and areas of human activity. Our approach outperformed prior work for distributed control of UAVs for wildfire tracking as well

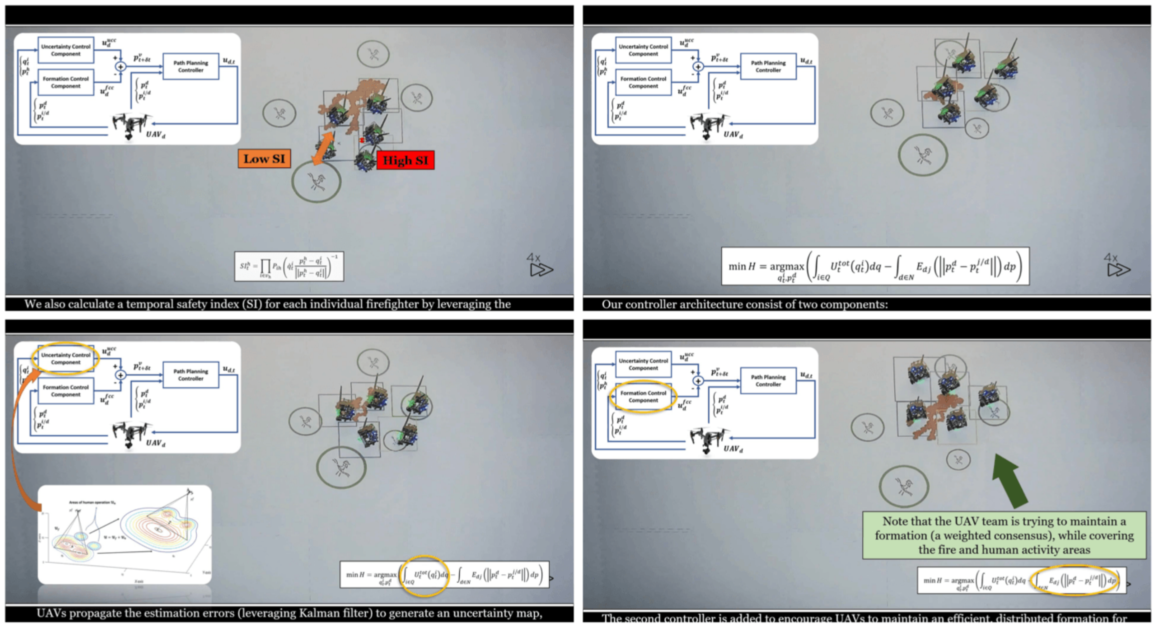

Low SI
High SI
Uncertainty Control Component
Formation Control Component
Path Planning Controller
We also calculate a temporal safety index (SI) for each individual firefighter by leveraging the
Our controller architecture consist of two components:
UAVs propagate the estimation errors (leveraging Kalman filter) to generate an uncertainty map,
Note that the UAV team is trying to maintain a formation (a weighted consensus), while covering the fire and human activity areas
The second controller is added to encourage UAVs to maintain an efficient, distributed formation for

Figure 4.7: Feasibility of the proposed distributed control algorithm for wildfire coverage evaluated on physical robots in Robotarium platform [151]. The experiment footage can be found at `https://youtu.be/j3YdIO5u_fE`.

as a reinforcement learning baseline.

4.2 Adaptive Leader-Follower Control for Multi-Robot Teams with Uncertain Network Structure

Traditionally-designed, centralized or decentralized control architectures typically rely on the availability of communication channels between neighboring robots as well as a known, static network structure to tightly coordinate their actions in order to achieve global consensus. Unfortunately, communication constraints and network disconnectivity are key bottlenecks in such approaches, leading to the failure of conventional centralized or decentralized networked controllers in achieving stability and global consensus. To overcome these limitations, we develop a centralized, coordinated-control structure for multi-robot teams with uncertain network structure. Our novel approach enables multi-robot teams to achieve consensus even with *disconnected* communication graphs. Leveraging model reference adaptive control framework and networked control architectures, we develop a coordinated leader-follower consensus controller capable of overcoming communication losses within the team, handling non-communicative robots, and compensating for environmental noise. We prove the stability of our controller and empirically validate our approach by analyzing the effects of reference graph structures and environmental noise on team performance. Finally, we demonstrate our novel controller in a multi-robot testbed.

4.2.1 Introduction and Motivation

Multi-robot systems have grown in importance during the past decade in cooperative task settings and for solving complex, large-scale problems that are intractable for individual robots. Multi-agent robotic systems have been applied to a wide array of applications, including environmental surveillance [5, 152, 8] and multi-robot formation control and object transport in automated factories and warehouses [153, 154, 155, 135, 156]. In many of the aforementioned scenarios, robots are typically required to navigate and maintain a specific, organized formation, e.g. to maintain a communication network [157, 158], to cooperatively survey an area [5], or to manipulate an object [153, 154]. While the need for developing multi-robot systems that are capable of autonomously coordinating and performing their tasks is critical, effective team coordination in these systems is challenging due to problems such as inter-agent communication constraints [17, 159].

Coordination of multi-robot teams typically entails a shared, common objective requiring the team to take actions according to the mutual interest(s) of the group [17]. In control theory, multi-robot control and coordination approaches can be divided into three main categories: (1) centralized control, (2) decentralized (distributed) control and, (3) hybrid-hierarchical (also referred to as semi-centralized [154]) control. Centralized methods consist of a central agent (e.g., a robot or a computer) that has access to global state information and oversees robots' navigation towards a desired location or formation [153, 160, 158]. On the other extreme, decentralized control and coordination methods do not include a central agent and each robot is taking actions solely based on its own local information [5, 161, 162, 163]. In hybrid-hierarchical control architectures, there exists a central planning agent which distributes tasks, while each robot then executes its actions locally [154, 8, 164].

Centralized coordinated-control approaches are typically criticized for burdensome communication overhead and are said to be memory-expensive in large-scale multi-robot systems [165]. As such, complex decentralized architectures, such as consensus-based networked control strategies, have been proposed to deal with the communication costs in large-scale multi-robot systems [166, 148, 167]. Nevertheless, centralized (and semi-centralized) architectures are still preferred in many, small-scale applications since decentralized methods suffer from issues such as sub-optimal solutions, rising from local, uncoordinated actions and computational complexity [165]. Moreover, it has been shown in prior work that existence of communication channels between all neighboring robots, such that a connected graph structure is formed and preserved, is a necessity to improve the quality of decentralized solutions [166, 148, 167]. However, a connected communication graph may not always be achievable, and a robot network might become disconnected, due to environmental constraints or hardware (e.g., wireless devices) limitations and failures. We note that this issue relates to both centralized and decentralized control architectures, as a consequence of which, these controllers become unstable and result in an inability to achieve global consensus under communication uncertainty and network disconnectivity.

Contributions

In this paper, we overcome these limitations in prior work by proposing a coordinated-control framework, suitable for centralized and semi-centralized multi-robot teams with uncertain communication network structures. Inspired by the adaptive control theory and the networked control systems, we develop a coordinated controller that takes advantage of the best of both centralized and decentralized control architectures while minimizing the aforementioned issues of each. Our proposed coordinated-controller can achieve team convergence even for a network of robots with a disconnected communication graph. To this end, we derive an adaptive control law for a leader-follower networked system that provably converges to mimic a desired network structure even though the real communication topology remains unknown to the robot agents (both leaders and followers alike).

In our approach, the team's communication network may be disconnected and its links may have varying strengths, with the only assumption that these communication links are not created/destroyed suddenly *during* the mission (i.e., an impulse). With this assumption, we avoid the need for Neural Network (NN) weight approximation as proposed in prior works [168, 169, 170], while reducing the required communication overhead due to a centralized model reference. Our motivation is to develop an elegant, robust method without relying on complex NN weight approximation, and to modify the conventional networked control strategy for multi-robot systems in scenarios where centralized controllers are feasible, but full inter-robot communication might be a concern. Moreover, our low-complexity coordinated adaptive controller can be used in online applications and is capable of handling non-communicative robots while also compensating for environmental noise.

We provide Lyapunov-based formal proofs of stability, and evaluate our controller by analyzing the effects of choosing different reference graph structures and environmental noises on the performance of robot team for leader-follower navigation. We empirically evaluate our algorithm on a set of coordinated control scenarios showing that our approach is able to quickly converge and achieve consensus whereas a traditional approach consistently fails. Finally, we demonstrate our approach on a physical, multi-robot testbed showing the feasibility and success of our approach.

4.2.2 Problem Statement and Formulation

To properly describe the problem under consideration, we choose a graph theoretic formulation of the robot network. Let us consider a collection of n agents with coordinates $z_i \in \mathbb{R}^d$ for $i = 1, \dots, n$. Assume that each agent, i, can communicate with a neighboring agent, j, only if agent j is within the communication range of agent i. This network is as an undirected graph $G = \langle V, E \rangle$, where $V = \{v_1, \cdots, v_n\}$ are vertices corresponding to each individual agent and $E = \{(v_i, v_j)\} \subset V \times V$ are edges, describing the connection (e.g., existence of communication channels) between robots i and j. These communication channels can be encoded by the adjacency matrix, $A = [a_{ij}]$, with $a_{ij} > 0$ if $(v_j, v_i) \in E$ and $a_{ij} = 0$ otherwise. Let d_i be the degree of vertex i, defined as the sum of elements in the i-th row (or i-th column due to symmetry) of A. By calculating the degree of all agents in the graph, we obtain the degree matrix, $D = \text{diag}(d_i)$, and the Laplacian matrix, $L = D - A$. It is known that $L = L^T \geq 0$ and any undirected connected graph G has the property that its first eigenvalue is zero while the rest are strictly positive [148].

Considering the aforementioned notions, the basic node-level consensus equation for a team of robots is given in Equation 4.38, in which agents simply move in the direction of the negative gradient of the total displacement error. Applying Equation 4.38 to a network of robots will result in all agents converging to the same location (e.g., reaching consensus).

$$\dot{z}_i = -\sum_{\langle j,i\rangle \in E} (z_i - z_j) \tag{4.38}$$

The ensemble-level dynamics of the robot network and accordingly, the description of leader-follower robot networks is presented in subsubsection 4.2.3.

4.2.3 Adaptive Leader-Follower Controller

This section formulates the proposed coordinated, adaptive, leader-follower controller. We begin by providing a detailed description of the leader–follower networks and then present our proposed adaptive leader-follower control structure. Next, we provide formal guarantees of stability and consensus for our robot network.

Leader-Follower Networks

Consider the following dynamical description of a leader-follower robot network at the ensemble-level for uni-dimensional, static, undirected graph $G = \langle V, E\rangle$ with n agents and m edges, as presented in Equation 4.39.

$$\dot{z} = u \tag{4.39}$$

In Equation 4.39, $z = \left[\zeta, x_n\right]^T \in \Re^n$ is the state vector in which $\zeta = [x_1, \cdots, x_{n-1}]^T \in \Re^{n-1}$ denotes the vector of follower coordinates (agents) and $x_n \in \Re$ represents a static leader (anchor node or desired location for agents to move towards). Moreover, $u = \left[u_\zeta, 0\right]^T \in \Re^n$ is the control signal in which $u_\zeta = \left[u_{\zeta_1}, \cdots, u_{\zeta_{n-1}}\right]^T \in \Re^{n-1}$ is the control input vector of the followers. We define the control input as in Equation 4.40.

$$u = -Lz \tag{4.40}$$

Next, we obtain the ensemble-level dynamics as in Equation 4.41.

$$\dot{z} = -Lz \tag{4.41}$$

In Equation 4.41, $L \in \Re^{n\times n}$ is the actual Laplacian matrix, meaning that it is unknown to the controller, and can be partitioned as represented in Equation 4.42 in which $L_f \in \Re^{n-1\times n-1}$, $v \in \Re^{n-1}$, and $b \in \Re$. Note that, if G is connected, then $L_f > 0$ [166].

$$L = \begin{bmatrix} L_f & v \\ v^T & b \end{bmatrix} \tag{4.42}$$

Accordingly, the followers dynamics along with the d-dimensional version of the problem can be derived as in Equation 4.43 and Equation 4.44, respectively, where $\xi = \left[\zeta_1, \cdots, \zeta_d\right]^T \in \Re^{(n-1)d}$ and $x_n = \left[x_{n_1}, \cdots, x_{n_d}\right]^T \in \Re^d$. Moreover, $\overline{L}_f = I_d \otimes L_f \in \Re^{(n-1)d\times(n-1)d}$, $\overline{v} = I_d \otimes v \in \Re^{(n-1)d\times d}$, and the operator $\otimes$ represents the Kronecker tensor product.

$$\dot{\zeta} = -L_f\zeta - vx_n \tag{4.43}$$

$$\dot{\xi} = -\overline{L}_f\xi - \overline{v}x_n \tag{4.44}$$

Applying the above distributed, *leader-follower* controller in Equation 4.44 to a team of networked-robots will drive the follower agents to the position of leader(s) [166]. We note that in our formulation, the disconnected sub-graphs do not necessarily need to have the same structure. We empirically validate this in our demonstrations (subsection 4.2.4-subsection 5.2.6).

Controller Formulation

Our objective is to derive a feedback control such that the consensus of the followers to the leader positions is achieved even if the network connectivity and structure are uncertain. For this purpose, we begin by defining the following reference model,

$$\dot{\xi}_m = -\overline{L}_{f_m}\xi_m - \overline{v}_m x_n \tag{4.45}$$

In Equation 4.45, $\xi_m = \left[\zeta_{m_1}, ..., \zeta_{m_d}\right]^T \in \Re^{(n-1)d}$, $\overline{L}_{f_m} = I_d \otimes L_{f_m} \in \Re^{(n-1)d\times(n-1)d}$, and $\overline{v}_m = I_d \otimes v_m \in \Re^{(n-1)d\times d}$. Note that $\overline{L}_{f_m}$ is the desired Laplacian and is a positive definite matrix (as the desired graph is connected). Thus, $H_m = -\overline{L}_{f_m}$ is Hurwitz and satisfies the Lyapunov equation for $Q \in \Re^{(n-1)d\times(n-1)d} > 0$ such that Equation 4.46 holds.

$$H_m^T P + P H_m = -Q \tag{4.46}$$

Equation 4.46 ensures that, for any Hurwitz matrix, H_m, and any positive definite matrix, Q, the solution, $P \in \Re^{(n-1)d\times(n-1)d}$, is always a positive definite matrix. We utilize the solution, P, in our adaptation mechanisms (Equation 4.55 and Equation 4.56), and the Lyapunov function (Equation 4.46) to guarantee the system stability.

Now, let us define the following direct adaptive model reference control law in Equation 4.47, where $u_\xi \in \Re^{(n-1)d}$ is the vector of the followers control inputs, $K_x \in \Re^{(n-1)d\times(n-1)d}$, and $K_r \in \Re^{(n-1)d\times d}$ are design matrices. Proper adaptation mechanisms will be designed later to update the control gains K_x and K_r (Equation 4.55 and Equation 4.56).

$$u_\xi = -\overline{L}_f\xi - \overline{v}x_n - K_x\xi - K_r x_n \tag{4.47}$$

By substituting Equation 4.47 into the followers dynamics $\dot{\xi} = u_\xi$, we arrive at the closed-loop system as follows in Equation 4.48.

$$\dot{\xi} = -\overline{L}_f\xi - \overline{v}x_n - K_x\xi - K_r x_n \tag{4.48}$$

Accordingly, to match the above closed-loop system in Equation 4.48 with the reference model in Equation 4.45, we assume that K_x^* and K_r^* are the ideal gains, leading to the *matching conditions* defined as follows in Equation 4.49 and Equation 4.50.

$$K_x^* = \overline{L}_{f_m} - \overline{L}_f \tag{4.49}$$

$$K_r^* = \overline{v}_m - \overline{v} \tag{4.50}$$

Now, we define the reference model tracking error, $e = \xi - \xi_m$, and the gain estimation errors, $\tilde{K}_x = K_x - K_x^*$ and $\tilde{K}_r = K_r - K_r^*$, using which the error dynamics are derived as follows, leading to Equation 4.53.

$$\dot{e} = -\overline{L}_f \xi - \overline{v} x_n - K_x \xi - K_r x_n + \overline{L}_{f_m} \xi_m + \overline{v}_m x_n \tag{4.51}$$

$$= -\overline{L}_f \xi - \overline{v} x_n - (\tilde{K}_x + K_x^*)\xi - (\tilde{K}_r + K_r^*)x_n + \overline{L}_{f_m}(\xi - e) + \overline{v}_m x_n \tag{4.52}$$

$$= -\overline{L}_{f_m} e + (\overline{L}_{f_m} - \overline{L}_f - K_x^*)\xi + (\overline{v}_m - \overline{v} - K_r^*)x_n - \tilde{K}_x \xi - \tilde{K}_r x_n \tag{4.53}$$

Substituting the matching conditions in Equation 4.49 and Equation 4.50 into Equation 4.53, we can rewrite the error dynamics as stated in Equation 4.54.

$$\dot{e} = -\overline{L}_{f_m} e - \tilde{K}_x \xi - \tilde{K}_r x_n \tag{4.54}$$

Accordingly, we suggest the following adaptation laws in Equation 4.55 and Equation 4.56 with the positive definite adaptation convergence matrices $\gamma_x \in \Re^{(n-1)d \times (n-1)d} > 0$ and $\gamma_r \in \Re^{(n-1)d \times (n-1)d} > 0$, and utilize the control law in Equation 4.47 for the followers dynamics $\dot{\xi} = u_\xi$ to follow the reference model represented in Equation 4.45.

$$\dot{K}_x = \gamma_x P e \xi^T \tag{4.55}$$

$$\dot{K}_r = \gamma_r P e x_n^T \tag{4.56}$$

Consensus Analysis and Proof of Stability

To ensure that followers achieve consensus (e.g., converging to leader locations), we utilize the Lyapunov stability notion. We define the following candidate Lyapunov function in Equation 4.57, in which P is a positive definite matrix and solution to Equation 4.46.

$$V(e,\tilde{K}_x,\tilde{K}_r) = e^T Pe + \text{tr}\left(\tilde{K}_x^T \gamma_x^{-1}\tilde{K}_x\right) + \text{tr}\left(\tilde{K}_r^T \gamma_r^{-1}\tilde{K}_r\right) \tag{4.57}$$

We leverage *Barbalat's lemma* [171, 172], reproduced below in Lemma Theorem 1, for guaranteeing asymptotic stability of nonlinear systems, and then present Theorem Theorem 2 to guarantee the stability of the closed-loop system.

Lemma 1 (Barbalat's Lemma) *If a Lyapunov function $V(t,x)$ satisfies the three conditions such that: 1) $V(t,x)$ is lower-bounded, 2) $\dot{V}(t,x)$ is negative semi-definite, and 3) $\ddot{V}(t,x)$ is bounded, then $\dot{V}(t,x) \to 0$ as $t \to \infty$.*

Theorem 2 *Consider the leader-follower system introduced in Equation 4.44 with a disconnected network structure, the candidate Lyapunov function in Equation 4.57, the control law in Equation 4.47, and the adaptation mechanisms in Equation 4.55 and Equation 4.56. Under the matching conditions in Equation 4.49 and Equation 4.50, and the reference model in Equation 4.45, consensus of the follower agents is achieved asymptotically, i.e., $\xi \to \xi_m$ and the gain estimation errors, $\tilde{K}_x$ and $\tilde{K}_r$, are bounded for all $e(0)$, $\tilde{K}_x(0)$, and $\tilde{K}_r(0)$.*

Proof 1 To prove the stability of our system through Theorem Theorem 2 and Lemma Theorem 1, we start by taking the time derivative of our candidate Lyapunov function in Equation 4.57 along the system trajectory in Equation 4.54. We derive the following:

$$\dot{V} = 2e^T P\dot{e} + 2\text{tr}\left(\tilde{K}_x^T \gamma_x^{-1}\dot{K}_x\right) + 2\text{tr}\left(\tilde{K}_r^T \gamma_r^{-1}\dot{K}_r\right) \tag{4.58}$$

$$= 2e^T PH_m e - 2\text{tr}\left(e^T P\tilde{K}_x\xi\right) + 2\text{tr}\left(\tilde{K}_x^T \gamma_x^{-1}\dot{K}_x\right) - 2\text{tr}\left(e^T P\tilde{K}_r x_n\right) + 2\text{tr}\left(\tilde{K}_r^T \gamma_r^{-1}\dot{K}_r\right) \tag{4.59}$$

Using Equation 4.46 and the trace properties, we derive the following:

$$\dot{V} = -e^T Qe - 2\text{tr}\left(\tilde{K}_x \xi e^T P\right) + 2\text{tr}\left(\tilde{K}_x^T \gamma_x^{-1} \dot{K}_x\right) - 2\text{tr}\left(\tilde{K}_r x_n e^T P\right) + 2\text{tr}\left(\tilde{K}_r^T \gamma_r^{-1} \dot{K}_r\right) \tag{4.60}$$

$$= -e^T Qe - 2\text{tr}\left(\tilde{K}_x^T Pe\xi^T\right) + 2\text{tr}\left(\tilde{K}_x^T \gamma_x^{-1} \dot{K}_x\right) - 2\text{tr}\left(\tilde{K}_r^T Pex_n^T\right) + 2\text{tr}\left(\tilde{K}_r^T \gamma_r^{-1} \dot{K}_r\right) \tag{4.61}$$

which using the adaptation laws in Equation 4.55 - Equation 4.56, it reduces to

$$\dot{V} = -e^T Qe \tag{4.62}$$

According to Equation 4.62, $\dot{V}$ is negative-semi definite, and therefore, Barbalat's lemma may be utilized to show $\dot{V} \to 0$. Since V is lower-bounded and Equation 4.62 implies that V is upper-bounded, the system errors, e, $\tilde{K}_x$, and $\tilde{K}_r$, are all bounded. It follows that conditions (i) and (ii) of Barbalat's lemma are satisfied. To verify the last condition of Barbalat's lemma, we calculate the second derivative of V along the system in Equation 4.54 as follows in Equation 4.63.

$$\ddot{V} = -2e^T Q\dot{e} = 2\left(\overline{L}_{f_m} e + \tilde{K}_x \xi + \tilde{K}_r x_n\right) \tag{4.63}$$

Since e and the reference trajectory, ξ_m, are bounded, then the boundedness of ξ is ensured. This, coupled with the boundedness of $\tilde{K}_x$, $\tilde{K}_r$, and the leader coordinate x_n implies that $\ddot{V}$ is bounded, concluding that all premises in Barbalat's lemma are satisfied, which in turn proves that $\dot{V} \to 0$ as $t \to \infty$. Therefore, referring to Equation 4.62, it is then proven that $\xi \to \xi_m$. Taken altogether, under the proposed controller, it is formally guaranteed that the consensus of follower agents is achieved (e.g., convergence to leader positions) in the presence of network dissconnectivity while the estimation errors $\tilde{K}_x$ and $\tilde{K}_r$ are bounded.

Remark 1 (The Stability and safety Assumptions) ***(1) Stability Assumptions:*** *Under our* *proposed coordinated-controller, ξ converges to the desired formation, ξ_m, as $t \to \infty$, if the following condition holds: fixed communication weights, which implies no links between agents are created or destroyed suddenly during the missions (i.e., P is constant which follows that $\dot{P} = 0$). However, in some applications time-varying communication weights may also exit, for instance, due to environmental constraints and obstacles or hardware (sender-receiver) poor performance. Time-varying weights imply that communication links between robots are not suddenly created or removed, but they may vary in time and $P_t \in (0,1]$. The immediate result of time-varying communication weights is that $\dot{P}$ is non-zero and P is slow-varying ($||\dot{P}|| \leq \varepsilon$ with $\varepsilon > 0$). In this case, it can be proven (see [173] for a similar discussion) that the error is uniformly ultimately bounded (UUB), e.g., $||e|| \leq \delta(\varepsilon)$ as $t \to \infty$ (δ is a function of $||\dot{P}||$), and thus, the proposed coordinated-controller will be stable. Accordingly, ξ converges to a small neighborhood of the desired formation, ξ_m, as $t \to \infty$, for which the ultimate bound of the neighborhood depends on the scale of $||\dot{P}||$.*

(2) Safety Assumption: *As described, under the proposed coordinated controller, agents may be disconnected, meaning that safety barrier certificates for collision avoidance cannot be applied. Accordingly, while developing collision avoidance algorithms for non-communicative robot teams falls beyond the scope of our current study, we make the assumption that in the case of a safety-critical application, each robot is equipped with proper collision avoidance for safety. We rely on the rich literature on vision-based collision avoidance (VCA) methods (e.g.,see [174, 175]) and assume each robot is equipped with proper sensors and a VCA method to refrain from "in-flight" collisions.*

Reference Model and Decaying Noise Effects on Follower Agents' Ultimate Configuration

Reference Model In subsubsection 4.2.3, asymptotic convergence of the tracking error, e, was guaranteed, which implies that the follower agents converge to the reference trajectory. This section analyzes the effect of the reference model on follower agents' ultimate formation. Since it is guaranteed that $\xi \to \xi_m$, the problem reduces to see *where* will each agent converge to, given ξ_m. To investigate this problem, we note that the solution to Equation 4.45 with $L_{f_m} > 0$ will asymptotically reach the quasi-static equilibrium ξ_m^* where

$$-\overline{L}_{f_m}\xi_m^* - \overline{v}_m x_n = 0 \tag{4.64}$$

Solving Equation 4.64 for ξ_m^* yields

$$\xi_m^* = -\overline{L}_{f_m}^{-1}\overline{v}_m x_n. \tag{4.65}$$

Since from Theorem Theorem 2 we know that $\xi \to \xi_m$ as $t \to \infty$, we can transform Equation 4.65 into Equation 4.66.

$$\lim_{t\to\infty}\xi(t) = -\overline{L}_{f_m}^{-1}\overline{v}_m x_n, \tag{4.66}$$

Equation 4.66 implies that the selection of the reference model $(\overline{L}_{f_m}, \overline{v}_m)$ as well as the leaders' positions x_n collectively determine where the follower agents will converge to. We empirically evaluate the effects of these parameters on the performance of the follower agents in simulation, and provide results in subsubsection 4.2.4.

Decaying Noise In the presence of a decaying noise, $e^{-\alpha t}$, affecting the system with $\alpha > 0$, the solution of Equation 4.45 can be obtained as shown in Equation 4.67.

$$\begin{aligned}\xi(t) =& e^{-\overline{L}_{fm}t}\xi(0) - \overline{L}_{f_m}^{-1}\overline{v}_m x_n(1 - e^{-\overline{L}_{fm}t}) \\ &+ \int_0^t e^{-\overline{L}_{fm}(t-\tau)}\mathbf{1}e^{-\alpha t}d\tau\end{aligned} \tag{4.67}$$

Utilizing the property $e^{-\overline{L}_{fm}(t-\tau)}\mathbf{1} = \mathbf{1}$ for the above equation, where $\mathbf{1}$ is the all-ones vector, we arrive at the solution for the noisy system as in Equation 4.68.

$$\lim_{t\to\infty}\xi(t) = -\overline{L}_{f_m}^{-1}\overline{v}_m x_n + \mathbf{1}\frac{1}{\alpha} \tag{4.68}$$

Comparing Equation 4.68 with Equation 4.66 as the general solution for leader-follower systems, we conclude that under the decaying noise $e^{-\alpha t}$, the ultimate follower configuration is drifted by the positive constant $\frac{1}{\alpha}$, which can be compensated for.

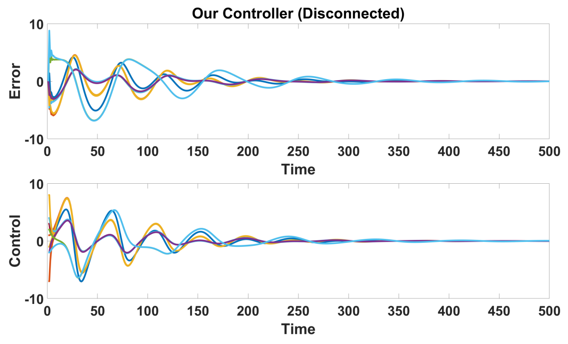
4.2.4 Empirical Evaluation

In this section, we empirically evaluate our coordinated adaptive controller to assess the coordination performance when the network is disconnected. First, we provide the simulation results in subsubsection 4.2.4 including error convergence results and adaptive gain assessments. We then present an empirical experiment on the effects of choosing the reference model (e.g., graph structure) and leaders' positions on the follower agents' ultimate configuration.

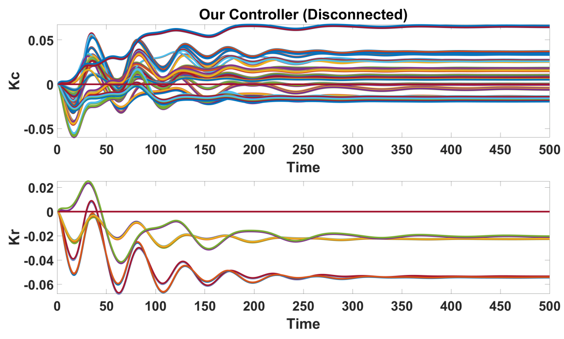
Empirical Stability and Convergence

To assess the performance of our proposed coordinated controller in presence of network disconnectivity, without loss of generality, we created an environment of ten agents (i.e., Dubins vehicles) as the follower robots that are separated into disjoint groups of size two, three and five. The follower agents within each group were connected to each other, but there was no communication channel between the agents of different groups. The goal was to navigate these agents to an area (e.g., for field coverage or to achieve a specific formation) and thus, we identified the coordinates of four vertices in a rectangular area as the leader (static) coordinates. Accordingly, we ran our proposed coordinated leader-follower controller on the collective set of ten agents by choosing a fully-sconnected graph between all agents as the reference communication model.

We compared our approach to a conventional networked-system controller on the same, disconnected graph that our algorithm has access to. To set an optimal – but not achievable – baseline, we also included a conventional leader-follower controller with a fully-connected graph. We note that we chose to compare our more-informed (e.g., centralized) adaptive,



(a) Tracking error and control effort



(b) Adaptation gains

Figure 4.8: This figure depicts the empirical stability and convergence results of our coordinated adaptive leader-follower controller with network disconnectivity. Figure 4.8a presents the error convergence and the control effort for each of the ten robot agents in the team (setup described in subsubsection 4.2.4), and Figure 4.8b shows the adaptive gains during the simulation.

networked controller with a regular distributed networked controller to demonstrate that the connectedness condition in such systems can be bypassed at the cost of a simple centralized model-reference. Our motivation was to modify the conventional networked control strategy for multi-robot systems in scenarios where centralized (or semi-centralized) controllers are feasible, but full inter-robot communication might be a concern.

The stability and convergence results for performance evaluation of our coordinated adaptive controller in the presence of network disconnectivity are presented in Fig. 1. Figure 4.8a depicts the tracking performance and control effort from which we see that all tracking errors and velocity control inputs converge to zero. This convergence is coincided with the moment that the adaptation gains converge to a bounded neighbourhood around their true values, as shown in Figure 4.8b. Figure 4.9 demonstrates the follower agents' behaviors in comparison with conventional controller, with and without network disconnectivity, in which black and blue lines show the positions of follower agents and leaders, respectively. It is observed that, in contrast with the conventional controller that fails navigating agents to the desired leader positions, the proposed adaptive controller can successfully performs the task in the presence of the network disconnectivity.

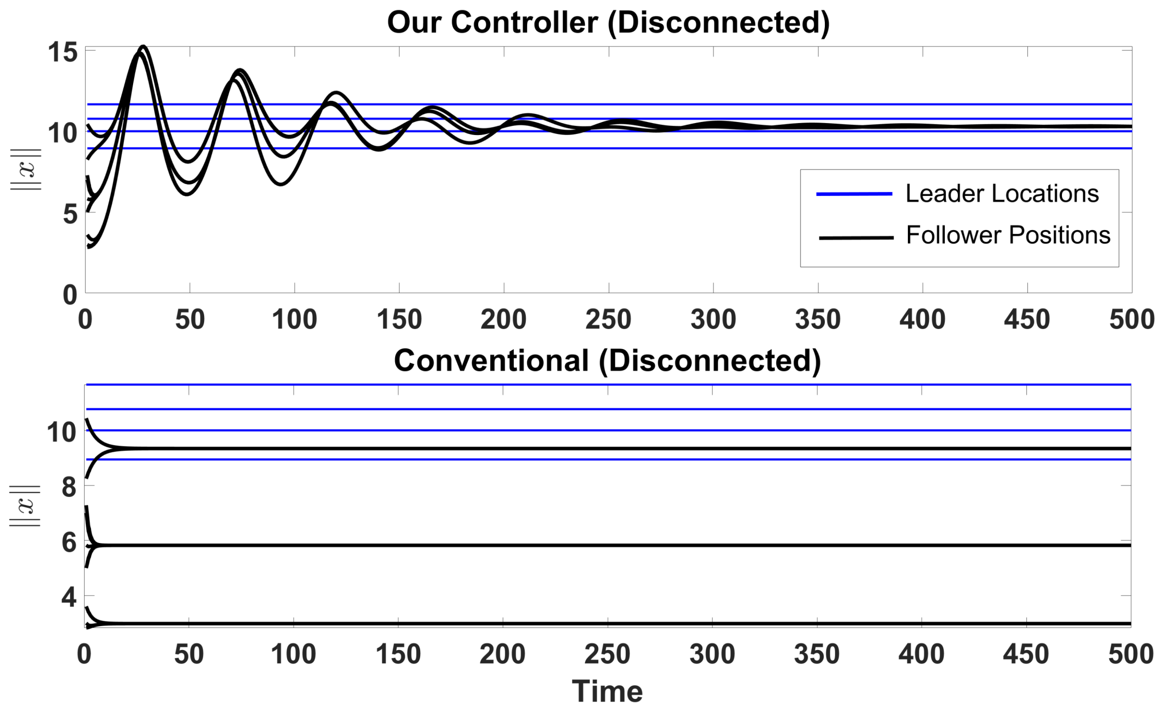
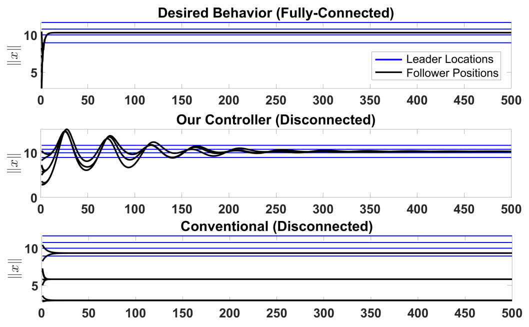


Figure 4.9: This figure demonstrates the follower agents' behaviors under our proposed coordinated adaptive framework in comparison with the conventional controller. The black and blue lines show the positions of followers and leaders, respectively.

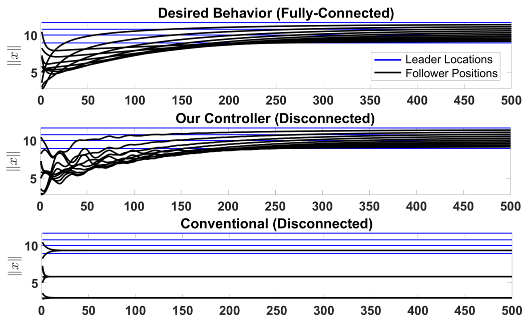
Effects of Reference Graph Structure

To evaluate the effects of choosing the reference communication graph on the performance of the team we examined three well-known graph structures: (1) complete graph, (2) cycle graph, and (3) linear graph. The experiment environment was similar to the setting in sub-subsection 4.2.4 and the resulting follower agents' behavior (e.g., position and convergence) are presented in 3 for complete (left), cycle (middle) and linear (right) graphs. As shown, our coordinated controller can successfully navigate the follower agents and mimic the reference communication model.

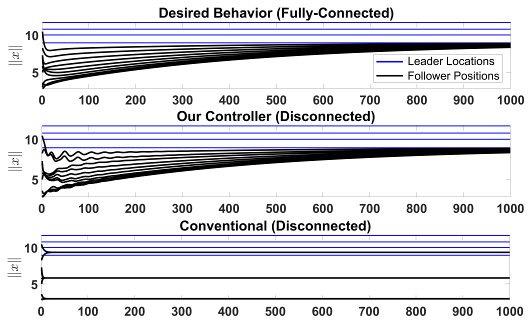
As shown in Figure 4.10a, in case of choosing a complete graph as the reference communication topology, the follower agents collectively converge to the mean of the four leader locations (blue lines) while for the other two cases followers are showing different behaviors. Figure 4.10b shows that follower agents are uniformly distributed between the smallest and largest leader locations in case of choosing a cycle graph as the reference model.



(a) Complete graph



(b) Cycle graph



(c) Linear graph

Figure 4.10: Effects of choosing reference communication graph model on the performance of the leader-follower controller and follower agents' behaviors (e.g., position and convergence) for complete (Figure 4.10a), cycle (Figure 4.10b) and linear (Figure 4.10c) graphs.

Moreover, Figure 4.10c shows that for a linear graph, followers converge to the smallest leader. Therefore, based on the underlaying application, one can choose the reference model to achieve a desired formation. For instance, for covering a large wildfire area, one can select the smaller and larger leader positions such that agents are distributed between the two locations, to cover the entire area. Moreover, we note that choosing the reference graph also affects the convergence rate of the followers to the leader positions. Accordingly, a weak communication graph (e.g., Linear) results in a slower convergence, as compared to stronger communication topologies.

4.2.5 Demonstration: Multi-Robot Testbed

To demonstrate the feasibility of our approach, we implemented and tested our coordinated adaptive leader-follower controller in the Robotarium, a multi-agent robotic platform [151]. The robot team in Robotarium is composed of a number of wheeled miniature, differential-

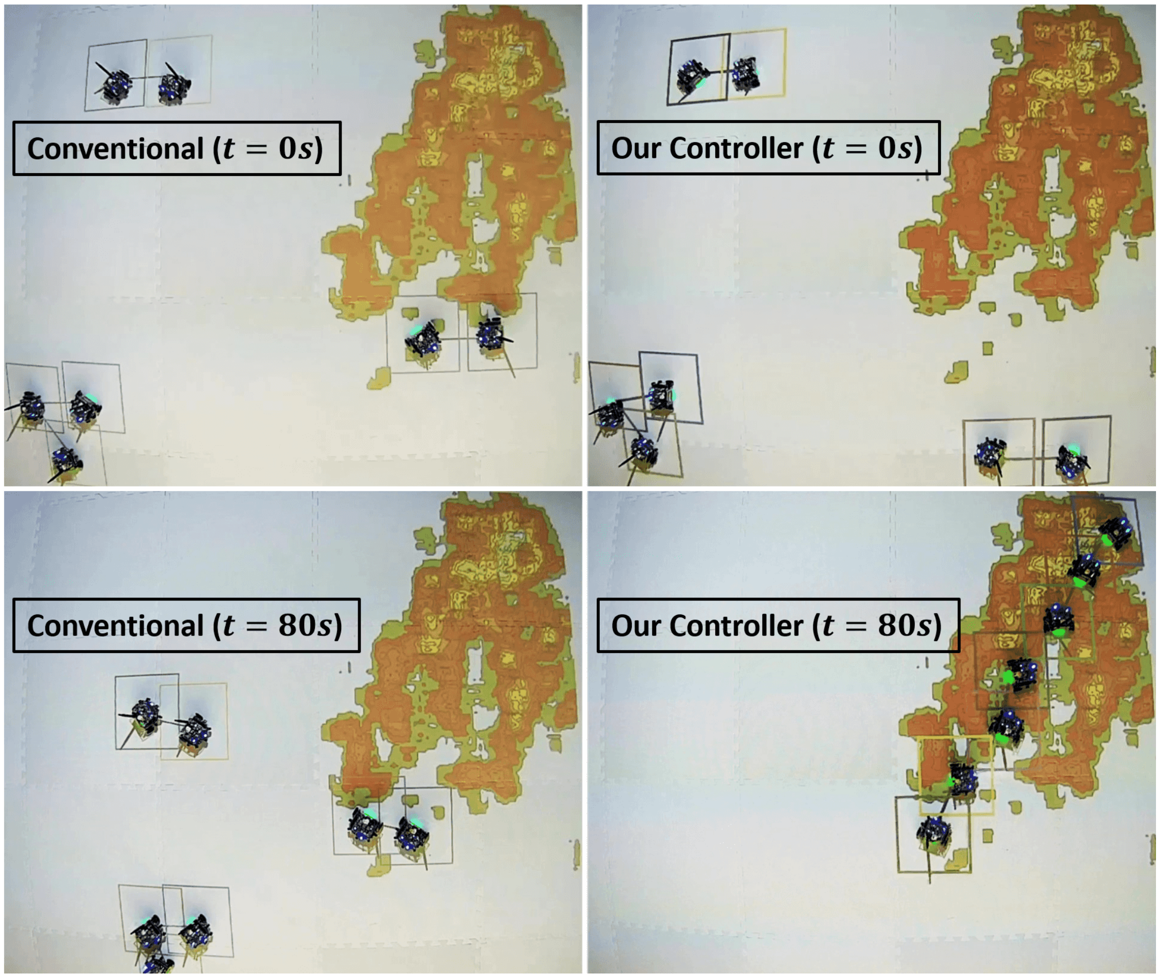


Figure 4.11: Initial ($t = 0s$) and final robot standings ($t = 80s$) in the field coverage experiment under both the conventional (left-side column) and our proposed controllers (right-side column). In our experiments, three disjoint groups (no communication between groups) of two, two, and three robots, were initialized. A cycle graph was selected as the reference communication model with leader locations set on lower-left and upper-right corners of the designated coverage area. (video link `https://youtu.be/mX8Sm-iH-kQ`).

drive robots which are tracked (i.e., position and orientation) in real-time by a webcam-based tracking system. For the experiment setup, seven robots that were separated in three disjoint groups of size two, two, and three, were initialized. Considering field coverage as the underlaying application, we selected a cycle graph as the reference communication model for our disconnected network of robots and smaller and larger leader locations were chosen to be the lower-left and upper-right corners of the designated coverage area so that robots cover the entire area (see discussion in subsubsection 4.2.4). Figure 4.11 presents initial and final robot standings in our experiments where squares around each robot shows the field-of-view (FOV) of the robot and solid lines between robots demonstrates communication channels. A similar experiment was performed under the conventional controller and results of both experiments are provided for comparison. The video recordings of experiments can be found on `https://youtu.be/mX8Sm-iH-kQ`.

4.2.6 Conclusion

In this paper, we developed an adaptive coordinated control strategy for the leader-follower systems with uncertain communication network structures. The proposed approach provides a novel coordinated controller to be applied to disconnected and/or non-communicative teams of robots with uncertain communication graph structure for which the conventional consensus-based networked controllers fail. We achieved this result by bypassing the connectedness condition at the cost of a simple centralized model-reference. Our motivation was to modify the conventional networked control strategy for multi-robot systems in scenarios where centralized (or semi-centralized) controllers are feasible, but full inter-robot communication might be an issue. Through the proposed controller, we take advantage of the best in both centralized and decentralized control architectures, while reducing the limitations of each. Simulation and experiments confirm the benefits of the proposed scheme over conventional methods that fail at achieving consensus. In future work, we aim to relax the assumption that no communication edges are added/removed during the mission and modify our controller to compensate for time-varying graphs.

CHAPTER 5
HIGH-LEVEL PERFORMANCE-GUARANTEED COORDINATED PLANNING FOR MULTI-ROBOT TEAMS

Multi-robot coordination approaches can be tackled at two separate levels: (1) high-level decision-making and (2) low-level control. The high-level decision-making module deals with planning a set of objectives (course of actions) among several possible options through which robot(s) can optimally (or at least satisfactorily) accomplish their task-objective. On the other hand, the low-level control module tackles the problem of designing appropriate control inputs for robot actuator(s) so that the robot(s) can follow a future trajectory as closely as possible. Our works in the previous chapter was focused on the low-level control.

In this chapter, we focus our studies on designing coordinated high-level plans for cooperative teams of robots. We tackle important aspects of multi-robot teaming such as providing performance-guaranteed service. We continue to use the aerial fire monitoring and wildfire fighting, introduced in section 3.1, as our running case-study.

In section 5.1, we study the problem of coordinated planning of a multi-UAV team for cooperative surveillance and tracking of a restless environment [7, 8]. We utilize a similar model-predictive mechanism as in section 4.1 to enable UAVs with the ability to reason about their plan for collaborative surveillance through actively estimating changing environment states. Particularly, we consider time-sensitive scenarios where only a limited number of UAVs are available for allocation. A central contribution of our work in section 5.1 is a set of analytical temporal and tracking-error bounds that allow UAVs to enable probabilistically-guaranteed coordination in tracking dynamic targets.

Next, in section 5.2, in an attempt to simultaneously tackle both the high-level planning and the low-level control stages of the coordination hierarchy, we develop an efficient hierarchical coordination framework for a composite robot team[1] composed of *perception*-only and *action*-only agents [9]. Our proposed framework in section 5.2 consists of two modules: (1) a MA-SARTSA algorithm under Partially Observable Semi-Markov Decision Process (POSMDP) as the high-level decision-making module to enable *perception* agents to learn to surveil in a restless environment with unknown number of dynamic targets and (2) a low-level coordinated control module that ensures probabilistically-guaranteed support for *action* agents.

5.1 Multi-UAV Planning for Cooperative Dynamic Field Coverage with Quality-of-Service Guarantees

In recent years, teams of robot and UAVs have been commissioned by researchers to enable accurate, online wildfire coverage and tracking. While the majority of prior work focuses on the coordination and control of such multi-robot systems, to date, these UAV teams have not been given the ability to reason about a fire's track (i.e., location and propagation dynamics) to provide performance guarantee over a time horizon. Motivated by the problem of aerial wildfire monitoring, we propose a predictive framework which enables cooperation in multi-UAV teams towards collaborative field coverage and fire tracking with probabilistic performance guarantee. Our approach enables UAVs to infer the latent fire propagation dynamics for time-extended coordination in safety-critical conditions. We derive a set of novel, analytical temporal, and tracking-error bounds to enable the UAV-team to distribute their limited resources and cover the entire fire area according to the case-specific estimated states and provide a probabilistic performance guarantee. Our results are not limited to the aerial wildfire monitoring case-study and are generally applicable to problems, such as search-and-rescue, target tracking and border patrol. We evaluate our approach in simulation and provide demonstrations of the proposed framework on a physical multi-robot testbed to account for real robot dynamics and restrictions. Our quantitative evaluations validate

[1] See subsection 3.1.2 for a detailed description of the composite robot teams

the performance of our method accumulating 7.5$\times$ and 9.0$\times$ smaller tracking-error than state-of-the-art model-based and reinforcement learning benchmarks, respectively.

5.1.1 Introduction

While multi-robot systems are capable of executing time-sensitive, complex missions that are intractable for a single robot, it is challenging to efficiently coordinate such systems and to optimize the collaborative behavior among robots [9, 12]. This coordinated planning problem becomes even more challenging when robots have to collaborate in a dynamic environment, such as in monitoring an active wildfire. Dynamic environments are restless; meaning regardless of robots' collective actions, the states of the environment continually change. As such, classical multi-agent planning and scheduling approaches designed for static environments will fail in such domains, since coordination plans made for a timestep may be inapplicable to the next step due to the environment dynamicity [176, 177].

In this work, we study the problem of coordinated planning of multi-UAV teams deployed as MSNs for cooperative surveillance and tracking of a dynamic environment. We enable UAVs with the ability to estimate the changing states of their environment and leverage these estimated information to reason about their cooperation plan for collaborative monitoring. Particularly, we consider safety-critical and time-sensitive scenarios where only a limited number of UAVs are available. Therefore, in our problem, not only it is important to reason robustly under environmental dynamicity via active planning with limited resources, it is exigent to have probabilistic guarantees that the UAV team can execute the coordinated plan. In our approach, the UAV agents actively evaluate a probabilistic error-bound that provides such guarantee and attempt to revise their coordinated plan when a performance guarantee cannot be provided due to changes in environment states. While the problem of multi-agent planning and scheduling has been studied extensively in prior work [176, 177, 178, 179, 180, 181], many of these works either lack the ability to actively plan in a dynamic environment, or do not provide a probabilistic performance guarantee that assures the teams' success in executing a coordinated plan.

Contributions

We develop a probabilistically-guaranteed framework for real-time, large-scale coordinated planning and cooperative coverage of dynamic environments. We ground our approach in application to aerial wildfire monitoring and develop an algorithmic framework in which we explicitly infer the latent fire propagation dynamics and leverage the estimated information in real-time to plan for a guaranteed quality of service. In our predictive mechanism, we define the term *service* as actively estimating the states of a propagating wildfire via limited number of UAVs and providing human firefighters on the ground with probabilistic guarantees regarding their proximity to a fire. We design our planning and coverage (i.e., monitoring) strategies upon a multi-step AEKF predictor and the FARSITE wildfire propagation mathematical model [122].

A primary motivation of our approach is the need for computationally lightweight, yet probabilistically-guaranteed, algorithms to scale to large numbers of UAVs and coverage areas. A central contribution of our work is a set of novel, analytical temporal and tracking-error residual bounds that allow UAVs to enable probabilistically-guaranteed coordination in monitoring and tracking dynamic targets (i.e., firespots). We derive these bounds in three different scenarios: (1) stationary fire, (2) moving fire without considerable grow (i.e., spawn), and (3) quickly moving, multiplying fire. For each case, we derive analytic and probabilistic temporal upper-bounds to facilitate performance guarantee for collaborative field coverage. The primary contributions of our work are:

1. An algorithmic planning framework based on Kalman estimation and uncertainty propagation to learn and leverage in real-time a predictive mechanism that enables probabilistic guarantees for tracking dynamic targets (i.e., firefronts).

2. Introducing the Uncertainty Residual Ratio (URR) tracking-error bound as well as a set

of novel, analytical temporal upper-bounds that allow UAVs to enable probabilistically-guaranteed coordination in monitoring and tracking dynamic targets (i.e., firespots)

3. An efficient and scalable collaborative field coverage algorithm by leveraging our analytical temporal bounds for centralized planning and distributed execution.

4. Presenting quantitative and experimental results in simulation and on physical robots that show the effectiveness of our URR bound and cooperative coordinated multi-UAV planning framework by demonstrating efficient scalability and significantly outperforming two prior SOTA baselines [36, 119] in a wildfire coverage and tracking task by accumulating $7.5\times$ and $9.0\times$ smaller tracking-errors, respectively.

We note that, despite the utility of UAVs as the robot agents and the aerial wildfire monitoring application in our work, the proposed coordinated, collaborative planning and area coverage algorithms as well as the derived set of analytical temporal and the URR tracking-error bounds are broadly applicable to other safety-critical applications that require mobile sensors to accurately track moving targets in their environment. See subsubsection 5.1.1 for a detailed discussion of the applicability of our proposed framework. We investigate the cooperative field monitoring problem by considering the wildfires in three separate scenarios of *stationary*, *moving* and *moving-spreading* target points (e.g., firespots).

In the application of wildfire fighting, human teams (i.e., firefighters) fight a wildfire until the fire is extinguished or working condition becomes too harsh to continue relative to the cost of abandoning. However, determining when a situation is too dangerous relative to the costs of abandoning the effort is a non-trivial ethical dilemma (e.g., how to measure the lives of immovable patients at a hospital in the fire's path relative to those of the firefighters) complicated by a lack of information regarding the fire's propagation characteristics. We take the perspective that this decision should remain in the hands of human professionals. As such, in our case-study, we focus 1) on developing a tight, probabilistic, upper-bound to reason about the minimum number of robots required to ensure high-quality information for the human decision-makers (subsection 5.1.4) and 2) measuring the tightness of our test by the number of robots required to satisfy this bound relative to other state-of-the-art approaches (subsection 5.1.7).

We empirically evaluate the performance, feasibility, and scalability of our framework in various experiments, alongside a state-of-the-art model-based [36] and a reinforcement learning (RL) benchmark [119] for UAV-based aerial field coverage and coordinated planning. Our experiments demonstrate a promising utility of our approach, accumulating 7.5$\times$ and 9.0$\times$ smaller tracking-errors than the two benchmark methods. Moreover, we assess the feasibility of our framework through implementation on physical robots in a mock wildfire monitoring scenario. The results of our experimental evaluations are presented in a supplementary video, available on `https://youtu.be/zTR07cKlwRw`.

General Applicability of the Proposed Framework

While grounded in an application to wildfire monitoring, the proposed coordinated planning and collaborative field coverage frameworks for teams of autonomous robots are broadly applicable to other domains and problems in which one is attempting to perform intelligent tracking and monitoring of many targets with few robotic sensors in large-scale operations [182]. Such applications include search-and-rescue [183, 184, 185], tracking of wildlife poachers [186, 187, 188], oil spill surveillance in oceans [189, 190, 152], border patrol and protection [191, 182] and even air traffic control for urban air mobility [192, 193, 194]. Here, we discuss these potential applications.

Search and Rescue Missions– Considering a scenario in which a natural disaster has struck a city, a team of UAVs can be used to keep monitoring various distant search sites for detecting survivors. Here, robots need to make sure to visit each search site frequently enough to not miss any signs of motion; here, our analytical temporal and measurement-uncertainty bounds for the case of static targets can provide a probabilistic guarantee for surveillance quality. This example can be extended to other search-and-rescue scenarios where humans (groups or individuals) are lost in forests or mountains and various sites need to be searched persistently or even for animal and wildlife monitoring and control, such as tracking specific animal species.

Border Patrol and Protection– Similar to the search-and-rescue example, border patrol and protection can be a relevant application for our framework in which aerial surveillance and tracking of multiple moving target (e.g., cars) can be performed by hovering UAVs. Various distant areas each including multiple targets can be monitored with a small team of UAVs which estimate the states (i.e., position and velocity) of specific targets. For this application, our analytical bounds for dynamic targets can provide the minimum number of UAVs, required to ensure a high tracking quality of targets. A related example would be the tracking of wildlife poachers.

Oil Spill Detection and Surveillance– Detecting and surveilling oil spills in the oceans are also a closely relevant application for our frameworks. Similar to the aerial wildfire detection and monitoring, teams of autonomous UAVs can be deployed for large-scale oil spill surveillance, including multiple distant and dynamically growing spills, similar to the third case of our wildfire case-study, the moving-spreading fire areas.

5.1.2 Related Work

Along the line of our work in the section, Bailon-Ruiz *et al.* [51] proposed a model-based planning algorithms to monitor a propagating wildfire using a fleet of UAVs. The approach tailors a variable neighborhood search to plan surveillance trajectories for a fleet of fixed-wing aircrafts according to a given fire propagation model and a given wind model. The fire state predictions are performed for hours into the future by using the integrated models and are updated through aircraft observations. Despite the similarities to our work, however, we did not consider the approach in [51] as a baseline for comparison, since the method assumes both a fire propagation model and a wind model to be given for making predictions and plans on the scale of hours into future and is designed for fixed-wing aircrafts with constrained motion dynamics. In our approach, we enable UAVs to actively infer fire propagation model parameters through environment observations and plan accordingly, whereas, in [51], the state estimation is performed for extended temporal windows into the future. An active state estimation framework may be able to better cope with changes in fire dynamics over time. In addition, the method in [51] directly plans trajectories for fixed-wing UAVs with constrained differential-drive dynamics which are not applicable to our presumed omni-directional UAVs. The focus of our method, on the other hand, is to provide a performance-guaranteed plan and an upper-bound on the number of UAV agents needed to monitor the fire areas without losing the track quality.

Furthermore, data-driven and learning-based approaches such as Reinforcement Learning (RL), have also been broadly applied to the problem of dynamic field coverage to enable collaborative monitoring of wildfires [119, 195, 196, 54, 197, 198]. The problems of high-dimensional state-space and imperfect sensory information, which are common in the aerial wildfire monitoring application, are tackled in [141] and [199] for teams of fixed-wing aircrafts using two deep RL methods. A similar problem was investigated and tackled in [200] where a Q-learning [126] was leveraged to learn one surveillance policy for a group of independent Q-learners. Ure *et al.* [201] proposed a decentralized approach for the problem of multiple learning and collaborating agents in fire monitoring cases where agents estimate different models from their local observations, but they can share information by communicating model parameters. In [13, 10], a graph-based actor-critic [126] method is introduced to learn efficient communication protocols for a group of cooperating heterogeneous agents (i.e., sensing and manipulating agents) performing wildfire fighting.

While RL-based approaches occasionally show promising results in specific contexts, such as for small-scale fires [119, 141], there are key limitations in terms of lack of formal guarantees on the boundedness of error, non-standard reward-function specification, scalability and inadaptability to domain shift, which are all limiting factors in application to safety-critical domains such as wildfire fighting.

5.1.3 Problem Formulation and Algorithmic Overview

Problem Statement

The proposed coordinated planning and collaborative coverage framework with quality-of-service guarantees is described and formulated in this section and is presented in Algorithm algorithm 2. We assume that firefront loci are detected through vision or thermal cameras [25, 26]. Moreover, as discussed in Figure 3.1a and Figure 3.1b, the UAV team is required to collectively monitor and track the fire (or generally moving targets) within several disjoint areas, such as areas of human-activity or human-defined areas of priority where high-quality information is indeed required. As such, we also assume that locations of these areas are known a priori.

Upon receiving the request to monitor and track the fire in a total of N_h areas of wildfire, an available UAV, d, from a set of homogeneous UAVs, $d \in \{\text{UAV}\}^{N_d}$ (N_d is total number of UAVs), is required to provide, online (or at within reasonable time frames) and high-quality information regarding the firefronts in the specified areas. The information here is referring to the estimated state information of a firespot, such as its location and velocity. To this end, the UAV d conducts a tour on firespots, $\left(q_t^1, \cdots, q_t^{N_q}\right) \in \{Q_t\}^{N_q}$, where N_q is the total number of detected firespots in all of N_h areas. This tour is performed to generate a graph, G_t, on firespots and accordingly a path, $\mathscr{P}_t^g$, on this graph for the UAV to follow as its tour-paths. The graph, G_t, is created by considering the firespots, $q_t^{N_q}$, as its nodes and the straight path between them as its edges. The path, $\mathscr{P}_t^g$, must be created and updated such that the UAV, d, is capable of visiting all N_h areas frequently enough such that the estimated state information, i.e., firespot locations and velocities, would remain up-to-date between the UAV visits. A list of inputs and outputs are summarized in Algorithm algorithm 2.

Since the velocity of the assigned UAV is limited to v_{max}^d and fire's propagation parameters (i.e., R_t, U_t and θ_t in FARSITE model in Equation 3.1) alter stochastically over time, it will be challenging for a single UAV to monitor all firespots within all N_h areas without

Table 5.1: Summary of key nomenclature used in our paper.

Notation	**Domain**	**Definition and Properties**
N_q	$\mathbb{N}^+$	Total number of detected firespots (or "targets" in general)
N_d	$\mathbb{N}^+$	Total number of available UAVs (or "robots" in general)
N_h	$\mathbb{N}^+$	Total number of areas to be monitored and tracked
$\hat{}$	–	Accent used for estimated variables
q_t^i	$\mathbb{R}^{1\times 2}$	Location of i-th firespot at time t (i.e., $q_t = \left[q_t^x, q_t^y\right]$)
p_t^d	$\mathbb{R}^{1\times 3}$	Position of d-th UAV at time t (i.e., $p_t = \left[p_t^x, p_t^y, p_t^z\right]$)
v_{max}^d	$\mathbb{R}$	Maximum linear velocity of UAV d
$\mathscr{M}_t$	$\mathbb{R}$	Fire propagation model (or in general, a target's motion model)
$\mathscr{O}_t$	$\mathbb{R}$	UAV's observation model (value at time t)
T_{UB}	$\mathbb{R}$	Upper-bound time it takes a UAV to complete a tour in an area
URR_t^q	$\mathbb{R}$	The uncertainty residual ratio
$\text{S}_{t\|t}$	$\mathbb{R}^{m\times m}$	Measurement residual covariance matrix; m is # state variables
$\mathscr{A}_t^d$	$\mathbb{N}^{1\times 2}$	Assigned UAV-path pairs

losing the track quality, particularly if the monitoring areas are distant. As such, more UAVs need to be recruited and the monitoring tasks need to be divided among them. Note that, UAV resources are limited and thus, in large-scale operations such as wildfire fighting it is not feasible nor is efficient to simply use as many UAVs as possible to increase the quality of the estimated information. Accordingly, the number of required UAV agents need to be systematically bounded to efficiently monitor fire propagation within all N_h human-defined areas of priority, while not losing the fire tracking quality. For this purpose, we propose leveraging online inference of wildfire dynamics and quantifying UAVs' accumulated estimation uncertainties about the fire model's parameters in order to efficiently coordinate the UAV team. For the readers' convenience, we provide Table 5.2 listing the key variables and notations used throughout the article.

Algorithmic Overview

Our solution to the aforementioned problem in subsubsection 5.1.3 is overviewed here. Algorithm algorithm 2 presents our multi-UAV coordinated planning framework for active monitoring of dynamic environments, such as wildfire areas. Figure 5.1 represents a flowchart diagram of the core analytical process for guaranteeing performance in our proposed algorithm.

Initially areas that need to be monitored are prioritized and divided from the rest of the map (Line 3). A selected UAV travels to these areas and gathers sensing information by flying over firespots and creating a graph, G_t, with detected firespots as its vertices and a tour-path, $\mathscr{P}_t^g$, by connecting the vertices on the generated graph as its edges (Lines 4-5). The graph G_t is an undirected graph generated by connecting each point to its closest neighboring point (i.e., minimum spanning tree) and thus, the initial, quickly-generated graph, G_t, and tour-path, $\mathscr{P}_t^g$, are not efficient and therefore, need to be modified. In our approach, generating the revised graph G_t' and the path $\mathscr{P}_t^{g'}$ on detected firespots within specified areas includes leveraging a Close-Enough Traveling Salesman Problem (CE-TSP) step to account for multiple firespots observable within UAVs' Field-of-View (FOV) (Line 6). In CE-TSP, a UAV only needs to get "close enough" to firespots for state estimation rather than going to the exact location of each firespot. Next, the UAV must determine through a *feasibility test* that whether monitoring all dynamic firespots within the N_h areas can be guaranteed by taking the generated path, such that the track of none of the spots is lost.

The performed feasibility test (Lines 9-13) is based on an analytical, probabilistic bound on the tracking error, which evaluates the measurement uncertainty residual about the state estimate, $\hat{q}_t^i$, of a firespot, q_t^i, at time t, where $i = 1, \cdots, N_q$. The tracking-error residual bound which we refer to as $\text{URR}_t^{\hat{q}}$, is evaluated through determining an upper-bound on the time, T_{UB}, required to cover each firespot in the N_h human-defined vicinities of priority (Line 7). In our feasibility test, T_{UB} is compared to the maximum time allowable for each fire track

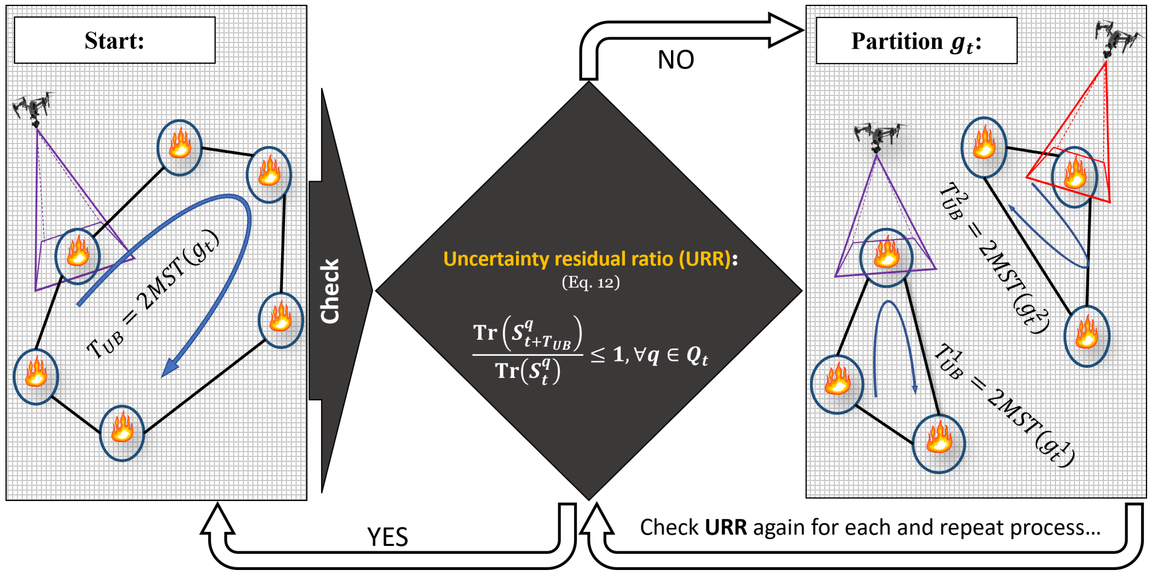

Start:
$T_{UB} = 2MST(g_t)$
Check
NO
Uncertainty residual ratio (URR):
(Eq. 12)
$\frac{\mathrm{Tr}\left(S^q_{t+T_{UB}}\right)}{\mathrm{Tr}(S^q_t)} \leq 1, \forall q \in Q_t$
Partition g_t:
$T^2_{UB} = 2MST(g^2_t)$
$T^1_{UB} = 2MST(g^1_t)$
YES
Check **URR** again for each and repeat process...

Figure 5.1: A flowchart diagram representing the URR checking process as the analytical safety condition. If the URR bound is not satisfied for a node in g_t, the graph is partitioned into two sub-graphs g^1_t and g^2_t (i.e., through k-means clustering), and accordingly into two paths, and another UAV is called-in from the available UAVs in the team. The process is repeated until URR is satisfied for all sub-graphs.

to propagate before measurement, so that the post-measurement residual is less than the residual after the last time a given firespot was observed (Line 35). As shown in Figure 5.1, if the test is satisfied, the aforementioned process continues and a near-optimal tour is computed via a k-opt procedure. If the test fails, the UAV divvies up the responsibilities for covering the fire locations by partitioning graph G'_t into $\langle g'_1, g'_2 \rangle$, recruiting another UAV to assist and repeating this process until URR$^{\hat{q}}_t$ bound is satisfied for all sub-graphs of G'_t. The partitioning process can be performed through clustering approaches such as k-means. UAVs' status are updated at the end for allocated/unallocated (i.e., 1/0 respectively) UAVs.

In the meantime, the unallocated team of UAVs can be utilized for coordinated, distributed field coverage (see subsection 5.1.5 and Algorithm algorithm 3), for monitoring the rest of the wildfire area (i.e., areas that are not specified or prioritized). To this end, unallocated UAVs generate a search graph and cluster the hotspots to generate a time-optimal path by leveraging the estimated temporal upper-bound T_{UB}.

Algorithm 2: Stages of the proposed multi-UAV coordinated dynamic field monitoring framework (pre-specified areas of priority) with guaranteed performance.

```
input  : Obtain the fire-map $\{Q_t\}^{N_q}$ and total number of areas to cover $N_h$, list of all UAVs
         $\{\text{UAV}\}^{N_d}$, set of UAV poses, velocities and observation model $\langle \{p_t\}^d, v^d_{max}, \mathscr{O}_t \rangle$, fire
         propagation model $\mathscr{M}_t$
output : Assigned UAV-path pairs $\{\mathscr{A}^d_t\}$
1  // main loop //
2  while MissionDuration do
3      Update fire-map and priority areas: $\langle \{Q_t\}^{N^1_q}, \{Q_t\}^{N^0_q} \rangle \leftarrow$ UpdateMap($\{Q_t\}^{N_q}$)
4      Estimate fire states and dynamics: $\{\hat{q}_t\} \leftarrow$ Sense($\{Q_t\}^{N_q}, \{p_t\}^d, \mathscr{M}_t, \mathscr{O}_t$)
5      Form a graph and compute path: $\langle \text{G}_t, \mathscr{P}^g_t \rangle \leftarrow$ ComputePath($\{\hat{q}_t\}$)
6      Modify graph of estimated spots: $\langle \text{G}'_t, \mathscr{P}^{g'}_t \rangle \leftarrow$ CETSP($G_t, \mathscr{P}^g_t$)
7      Compute $T_{UB}$ for $d \in \{\text{UAV}\}^d$ to cover $\text{G}'_t$: $T^{g'}_{UB} \leftarrow$ TUB($\mathscr{P}^{g'}_t, \{\hat{q}_t\}, v^d_{max}$)
8      Create a set from graphs and their corresponding $T_{UB}$s: $\{\mathscr{G}\} \leftarrow \langle \text{G}'_t, T^{g'}_{UB} \rangle$
9      while $\exists g \in \{\mathscr{G}\}$ s.t. False $\leftarrow$ FeasibilityTest($\{\mathscr{G}\}, \{\hat{q}_t\}$) do
10         $\langle \{g'_1, g'_2\}, \{\mathscr{P}'_{g_1}, \mathscr{P}'_{g_2}\} \rangle \leftarrow$ CETSP(Cluster($\{\mathscr{G}\}^g$, 2))
11         $\{T^{g'_1}_{UB}, T^{g'_2}_{UB}\} \leftarrow$ TUB($\{\mathscr{P}'_{g_1}, \mathscr{P}'_{g_2}\}, \{\hat{q}^g_t\} \in g, v^d_{max}$)
12         $\{\mathscr{G}\} \leftarrow \langle \{g'_1, g'_2\}, \{T^{g'_1}_{UB}, T^{g'_2}_{UB}\} \rangle$
13     end
14     Assign UAVs to paths: $\{\mathscr{A}^d_t\} = \langle \mathscr{P}^{g'}_t, d \rangle, \forall d \in \{\text{UAV}\}^{N_d}$
15 end
16 // inner functions //
17 def Sense($\{Q_t\}^{N_q}, \{p_t\}^d, \mathscr{M}_t, \mathscr{O}_t$):  // dynamic fire state estimations
18     $\hat{q}_{t+1} = \arg\max_{q_{t+1}} \rho \left( q_{t|t-1}, p_{t|t-1}, \mathscr{M}_t, \mathscr{O}_t \right)$
19 def ComputePath($\{\hat{q}_t\}$):  // generate and optimize path
20     $\text{G}_t, \mathscr{P}^g_t \leftarrow$ MST($\{\hat{q}_t\}$)  // MST(.): minimum spanning tree graph
21     while TimeAvailable do
22         $\mathscr{P}^g_t \leftarrow$ k-opt($G_t, \mathscr{P}^g_t$)  // k-opt(.): to optimize path
23     end
24 def TUB($\mathscr{P}^g_t, \{\hat{q}_t\}, v^d_{max}$):  // temporal service upper-bound
25     Determine fire scenario according to estimated states $\hat{q}^i_t$
26     switch
27         Case (1): $T^{C1}_{UB} \leftarrow \frac{2\text{len}(\mathscr{P}^g_t)}{v^d_{max}}$ // See Equation 5.8 for details
28         Case (2): $T^{C2}_{UB} \leftarrow \frac{8\zeta^\alpha(|\text{G}_t|-1)\text{len}(\mathscr{P}^g_t)}{v^d_{max}\left(1-4\zeta^\alpha(|\text{G}_t|-1)\right)}$ // See Equation 5.9 for details
29         Case (3): $T^{C3}_{UB} \leftarrow \frac{-\beta+\sqrt{\beta^2-4\gamma\delta}}{2\gamma}$ // See Equation 5.15 for details
30 def FeasibilityTest($\{\mathscr{G}\}, \{\hat{q}_t\}$):
31     if $\frac{\text{Tr}\left(S_{t+T_{UB}|t}\right)}{\text{Tr}\left(S_{t|t-1}\right)} \leq 1, \forall g \in \{\mathscr{G}\}$, return True else, return False
```

5.1.4 Coordinated Planning for Monitoring with Guaranteed Quality-of-Service

Our objective is to propose a performance-guaranteed coordinated planning framework for collaborative wildfire tracking. Particularly, our approach consists of a probabilistically-guaranteed, realtime coordination algorithm that provides human firefighters with real-time information about fire states within the specified areas. We then develop a predictive distributed coverage method for large-scale dynamic field coverage based on our probabilistic bounds to improve the performance in both wildfire area coverage and firefront tracking.

Online Inference of Wildfire Dynamics

The online inference of the wildfire mathematical model parameters through UAVs' environment observation is the same as in subsection 4.1.5. As such, in this section, we briefly summarize the process through leveraging the AEKF.

To perform a real-time inference of wildfire dynamics, we utilize the AEKF to predict a distribution over the states of observed firespots and accordingly, a measurement covariance for each firespot (i.e., each discretized point of firefront) through non-linear error propagation [202, 146]. In a linear system with Gaussian noise, the Kalman filter is optimal [203]. In a system that is nonlinear, the Kalman filter can be used for state estimation, but other methods such as the particle filter may give better results, normally at the price of significant additional computational effort [203]. In our work, we develop computationally lightweight algorithms to scale to large numbers of UAVs and coverage areas. As such, we leverage a Kalman filter as it provides such low-complexity algorithm while also performing satisfactorily.

Considering $q_t = \left[q_t^x, q_t^y\right]$ as the location of firefronts on the ground at time, t, and $p_t = \left[p_t^x, p_t^y, p_t^z\right]$ as the UAV pose, we seek to estimate a distribution over a firespot's location one step forward in time, $\hat{q}_{t+1}$, given the current firefront distribution and the previous measurement (shown as $q_{t|t-1}$ in our notations throughout the paper), fire propagation model with parameters at time, t, $\mathscr{M}_t$, and UAV observation model of the field, $\mathscr{O}_t$. Considering the parameters in fire propagation model (i.e., R_t, U_t and θ_t in FARSITE model in Equation 3.1), the firespot state estimation problem can be formulated as maximizing the joint probability density function (PDF), ρ, in Equation 5.1. Equation 5.1 describes a joint probability density function for estimating the next-step position of a firespot, given the current firefront distribution and the previous measurement, fire propagation model, and UAV observation model. Therefore, maximizing this PDF means finding a value of the random variable (the value estimate) that can occur with the highest probability.

$$\hat{q}_{t+1} = \underset{q_{t+1}}{\arg\max}\, \rho \left(q_{t|t-1}, p_{t|t-1}, \mathscr{M}_t \left(R_{t|t-1}, U_{t|t-1}, \theta_{t|t-1} \right), \mathscr{O}_t \left(q_{t|t-1}, p_{t|t-1} \right) \right) \tag{5.1}$$

The PDF in Equation 5.1 is calculated through the AEKF estimator. The process model, $\mathscr{M}_t$, predicts states given its parameters (i.e., R_t, U_t, θ_t) and the observation model, $\mathscr{O}_t$, maps the predicted state estimates to observation space.

Moreover, we incorporate process and observation noises to account for error induced by methodological uncertainties (i.e., fire propagation model and UAV observation model inaccuracies) and other sensor errors (i.e., camera and weather forecasting equipment), which are modeled as white noise with covariances Λ_t and Γ_t, respectively. We leverage an adaptive update framework [146] for process and observation noise covariances in standard EKF, which introduces *innovation*- and *residual*-based updates for Λ_t and Γ_t, respectively.

$$\Lambda_t = \gamma \Lambda_{t-1} + (1-\gamma) \left(K_t \tilde{d}_t \tilde{d}_t^T K_t^T \right) \tag{5.2}$$

$$\Gamma_t = \gamma \Gamma_{t-1} + (1-\gamma) \left(\tilde{y}_t \tilde{y}_t^T + H_t P_{t|t-1} H_t^T \right) \tag{5.3}$$

In above equations, γ is the update step-size, $\tilde{y}_t$ is the measurement residual, and $\tilde{d}_t$ is the measurement innovation. The measurement residual is defined as the difference between actual measurement and the predicted measurement using the posterior, while the measurement innovation is defined as the difference between the actual measurement and its predicted value. Moreover, $K_t = P_{t|t-1} H_t^T S_t^{-1}$ is the near-optimal Kalman gain, in which,

$P_{t|t-1}$ is the predicted covariance estimate, H_t is the observation Jacobian matrix and S_t is the covariance residual. Through Equation 5.2-Equation 5.3, Λ_t and Γ_t are adaptively estimated as the Kalman filter leverages its observations to improve the predicted covariance matrix, $P_{t|t-1}$. Accordingly, to derive the AEKF uncertainty (i.e., measurement-residual) propagation equations, we define the process state vector as $\vec{\Theta}_t = \left[q_t^x, q_t^y, p_t^x, p_t^y, p_t^z, R_t, U_t, \theta_t\right]^T$ and $\vec{\Phi}_t = \left[\varphi_t^x, \varphi_t^y, \hat{R}_t, \hat{U}_t, \hat{\theta}_t\right]^T$ as the mapping vector through which the state estimates are translated into a unified *angle*-parameters, as firespots on the ground are perceived by hovering UAVs (**??**) [5]. Eventually, the model and observation measurement uncertainties propagated by AEKF estimation can be shown as in Equation 5.40-Equation 5.41.

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + \Lambda_t \tag{5.4}$$

$$\mathrm{S}_{t|t} = H_t P_{t|t-1} H_t^T + \Gamma_t \tag{5.5}$$

Here, $P_{t|t-1}$ is the predicted covariance estimate and $\mathrm{S}_{t|t}$ is the innovation (or residual) covariance matrix. Moreover, F_t and H_t are the process and observation Jacobian matrices which include the gradients of FARSITE model equations in Equation 3.2-Equation 3.3 with respect to all state variables in $\vec{\Theta}_t$ and the gradients of UAV's observation model with respect to all observation space variables in $\vec{\Phi}_t$, respectively (see section 4.1 for derivations).

Guaranteeing the Quality of Service

Here, we first present the CE-TSP solution for generating more efficient tour-paths for agents. Next, we introduce and elaborate on our analytical condition for performance guarantee, i.e., the Uncertainty Residual Ratio (URR) bound, and then, we derive the analytical temporal upper-bounds for a probabilistically-guaranteed coordination based on URR.

Close-enough Traveling Salesman Problem (CE-TSP) When a UAV agent is tasked to monitor (i.e., estimate the firefront states and closely track each firespot), the first step in our coordinated planning module is to generate a search graph, G_t, with firefront points within the specified areas as the vertices and distances in between as the edges. To this end, we leverage the CE-TSP with Steiner zone [204] variable neighborhood search where the agent only gets "close enough" to each fire-point instead of visiting their exact locations[2].

In the application of aerial wildfire propagation monitoring, at each time-step, a UAV can observe multiple firespots (i.e., nodes) within its FOV simultaneously. As such, it is not required for the UAV to travel to the exact location of each firefront point. Instead, the UAV first identifies the overlapping Δ-disks between k firespots as their corresponding Steiner zone [204] and then chooses the centroid node from the identified Steiner areas as a new vertex in its search graph instead of the original k points. Through this process the original graph G_t is modified into a revised graph G'_t (Line 5 in Algorithm algorithm 2). The variable k can be tuned empirically and based on the background application.

Analytical Performance Guarantee Condition: Uncertainty Residual Ratio (URR) When the modified search graph, G'_t, is generated, the UAV performs tours on this graph by visiting each graph-node. Upon reaching a new node, the UAV updates the firespot's state estimate and calculates two quantities: (1) the analytical upper-bound time, T_{UB}, required to complete a tour on G'_t and arrive back at the current node considering the new state estimates (described in paragraph 5.1.4) and (2) the current measurement residual covariance through Equation 5.41. Leveraging these two parameters, the UAV propagates the uncertainty residual for T_{UB} steps into the future by repeatedly applying AEKF's prediction step (i.e., Equation 5.40) T_{UB} times, and then performing an update step at the end (i.e., Equation 5.41). In this way, we emulate gathering a measurement again and compute the innovation covariance at the end of the prediction steps. We introduce the Uncertainty

[2]Note that, this step might not be required in applications other than aerial wildfire monitoring, in which instead of N_h fire areas, N_h specific moving points/targets need to be monitored. Accordingly, the CE-TSP step in our framework can be replaced with a regular TSP [205].

Residual Ratio ($\mathrm{URR}_t^{\hat{q}}$) in Equation 5.6, in which Tr(.) represents the trace operation.

$$\mathrm{URR}_t^{\hat{q}} = \frac{\mathrm{Tr}\left(S_{t+T_{UB}|t}\right)}{\mathrm{Tr}\left(S_{t|t-1}\right)} \quad \text{and} \quad \mathrm{URR}_t^{\hat{q}} \leq 1, \forall q \in \{Q_t\}^{N_q} \tag{5.6}$$

The $\mathrm{URR}_t^{\hat{q}}$ bound is an indicator of the scale to which the UAV agent is capable of tracking the firespot, q_t, without losing tracking information, while performing tours on the search graph, G_t'. An unsatisfied URR bound (i.e., a URR value greater than one) indicates growing uncertainty and demonstrates a quickly growing (or propagating) wildfire, about which the UAV will not be capable of providing online state-estimates and without loosing any track information, while completing a tour on G_t'. Similarly, a URR smaller than one for all firespots in the current graph indicates that online information can be provided by the UAV while keeping track of all propagating firespots. We reiterate that, the goal of the UAV team here is to provide firefighters with the online, high-quality information about the propagating firefront within the specified areas such that these information can be used in real-time for strategizing firefighting plans.

To maintain control over the measurement uncertainty, we posit that the UAV observers would want the measurement uncertainty residual with respect to a target on the ground not to increase from $t = t_0$ to $t = t_0 + kT_{UB}$ for any positive integer constant k if the UAV observes the target from the same relative position. The reason is that the measurement uncertainty residual as computed by the EKF in our formulation is only dependent on the relative distance between the observer and the target and is independent of time (please refer to Appendix A for a mathematical proof). As such, the bound in URR ratio determines if the current uncertainty residual (the denominator of Equation 5.6) is greater or smaller than the uncertainty residual at time $t + t_{UB}$ (the numerator of Equation 5.6) when the UAV completes the tour and revisits the current target. Accordingly, if the ratio is greater than one, it means that the uncertainty residual is increasing, indicating the UAV is falling behind on monitoring the target and the tracking quality is getting worse.

As demonstrated in flowchart diagram in Figure 5.1, if the URR bound is not satisfied for a graph node ($\text{URR}_t^q > 1 \text{ for } q \in \{Q\}^{N_q}$), the generated search graph, G_t', is partitioned into two smaller sub-graphs $\langle g_1', g_2' \rangle$ (i.e., through k-means clustering), and accordingly two paths $\langle \mathscr{P}_{g_1}', \mathscr{P}_{g_2}' \rangle$. Therefore, the UAV divvies up the monitoring task into smaller portions and then recruits another UAV from the available UAVs in the team to collaborate by taking over one of the sub-graphs for monitoring. This process is repeated for all sub-graphs and all of their nodes until $\text{URR}_t^{\hat{q}}$ is satisfied for all $q \in \{Q_t\}^{N_q}$ and t. We note that, in our formulation, the UAVs only begin the surveillance and tracking the URR bound when they arrive to the determined sub-graph. As such, to minimize the time it takes a UAV to arrive to its assigned sub-graph we solve an optimization problem. Each UAV is assigned to one partition by solving a minimization problem with UAVs as variables, partitions as domains, and distance to the partition centroid as constraints. This way, the closest available UAV is selected for a sub-graph. Additionally, as described in paragraph 5.1.4, in cases where several firespots can be observed in a UAV's FOV at once, the URR bound is computed for the centroid node from the identified Steiner areas.

The introduced URR bound in Equation 5.6 depends on the upper-bound traverse time, T_{UB}, which itself is dependent on two major factors: (1) the maximum linear velocity of the UAV (v_{max}^d) and (2) the propagation rate of the fire (Or in general, the motion velocity of any dynamic target which is subject to monitoring). As such, the relations between T_{UB} and the aforementioned two factors need to be derived analytically for all possible scenarios. The following section is dedicated to analyzing such scenarios in the face of the aerial fire monitoring domain and the respective analytical temporal bounds are presented.

Probabilistic Temporal Upper-Bound for Service (T_{UB}) In this section, we derive a probabilistic upper-bound, T_{UB}, on the time required by a UAV to service (i.e., visit once and estimate states) each fire location. The T_{UB} is used in Equation 5.6 to determine whether a UAV can be probabilistically guaranteed to service each fire location fast enough to ensure

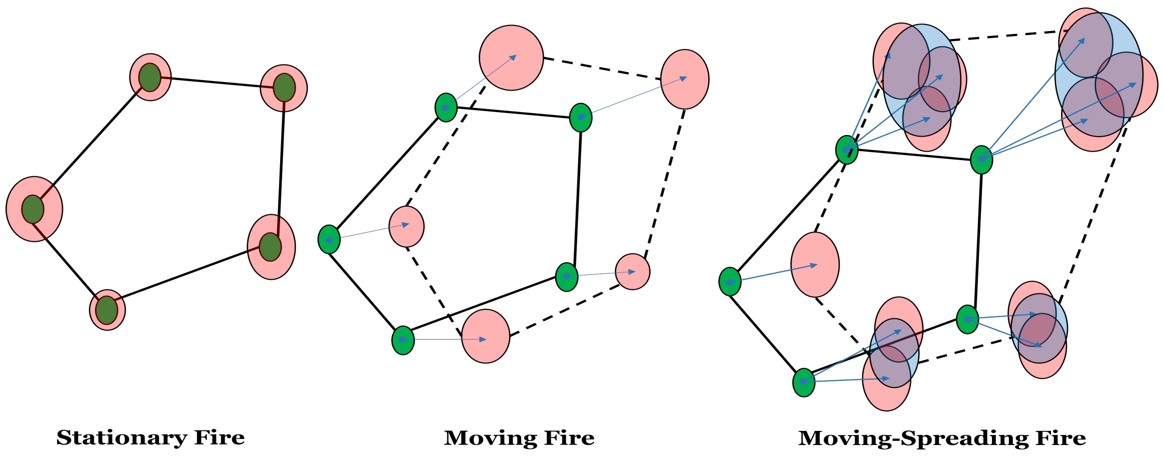


Figure 5.2: The three considered scenarios in which, a fire (i.e., target) could be stationary (left), moving (middle) or moving-and-spreading (right). Green dots show a firespot's current location and red circles represent the distribution over the firespot's next-step location. Blue circles show the Steiner zones. Note that the black, solid and dashed lines represent a UAV's current and next-step tour paths, respectively.

a bounded track residual for each firespot. As described in paragraph 5.1.4, the T_{UB} is dependent on two major factors: (1) the maximum linear velocity of the UAV (v^d_{max}) and (2) the propagation velocity and growth rate of firespots. As such, we derive three bounds, one for each of the following possible scenarios: (1) stationary target points[3], (2) moving target points, and (3) moving-spreading target points. Figure 5.2 depicts the three mentioned scenarios subject to our study.

For the derivations, we reason about the velocity of firespots (e.g., target points), ζ, at the α confidence level, such that, $Pr[\zeta < \zeta^{\alpha}] = 1 - \alpha$ where smaller α means more conservative. In other words, the probability of the fire being quicker than ζ^{α} is α. As such, we need to derive an α-dependent probability for our upper-bound service time T_{UB}, such that it is lower-bounded by the *actual maximum time, T^*, a UAV can go without visiting a $q \in \{Q\}^{N_q}$ and not losing its track quality*. Accordingly, and assuming conditional independency between firespots' velocities given latent fire dynamics[4], the probability that our bounds are "correct" (meaning that $T_{UB} \geq T^*$) can be shown as in Equation 5.7. For

[3]The scenario designs are motivated such that they expand the applicability of our framework to domains other than wildfire monitoring, and thus, here we use the term *target points* instead of *firespots*.

[4]We explicitly estimate the latent fire dynamics in our AEKF model.

our experiments, we set $\alpha = 0.05$. While, specific assumptions made for each Case are discussed in the respective Section, we note that the accuracy of the presented bounds depend on the level of accuracy of the utilized model for fire propagation[5], or generally for applications other than wildfire monitoring, the accuracy of the approximate motion model used for a moving target subject to monitoring.

$$Pr[T_{UB} \geq T^*] \leq 1-(1-\alpha)^{N_q} \tag{5.7}$$

Case 1: Stationary Target Points – In the first scenario all firespots are almost stationary (i.e., stationary costumers in the travelling salesman problem). In this case, the search graph, G_t', is a static graph, and therefore, we utilize a minimum spanning tree (MST) to obtain the initial path, $\mathscr{P}_t^{g'}$. The MST path (i.e., the Hamiltonian cycle computed from the MST) is not the optimal path, but it is fast to generate, and thus, it ensures a quick, sound way to obtain a path that leads to guaranteed service. Once an initial consensus is reached upon providing the updated firefront state estimates according to the MST path, the assigned UAVs can then spend time improving the efficiency of their tours through the k-opt algorithm. Accordingly, we derive the upper-bound time for a UAV with maximum linear velocity v_{max}^d to complete a tour on the static search graph in Case 1, as the Hamiltonian cycle of the generated MST path, shown by its spatial cost $\Delta\mathscr{T}^{\text{MST}}$ as introduced in Proposition 1. The $\Delta\mathscr{T}^{\text{MST}}$ is the length of the generated MST path.

Proposition 1 *The upper-bound time for a UAV to complete a monitoring tour, $T_{UB}^{(C1)}$, in Case 1, the stationary target points, can be calculated as shown in Equation 5.8.*

$$T_{UB}^{(C1)} = \frac{2}{v_{max}^d}\Delta\mathscr{T}^{MST} \tag{5.8}$$

[5]Here, we used the FARSITE model which can be replaced with any other parameterized fire propagation model, such as the correctable fire simulation model introduced in [123].

Case 1 is presented to develop a tight bound employed under nominal fire conditions and a foundation to derive the traverse temporal upper-bounds for the other cases. Moreover, while there are many approaches for approximating the TSP with guarantees, such as Christofides ($O(n^3)$), we choose other approaches with lower complexity, such as the double tree ($O(n^2)$) algorithm. This is because we consider large-scale scenarios with thousands of discrete firespots (discretization of hundreds of square-miles), which signifies the importance of the lower upper-bound on complexity.

Case 2: Moving Target Points – In the second scenario firespots, move but do not grow (i.e., spawn) considerably. As shown in the middle plot in Figure 5.2, the upper-bound service time, T_{UB}, for a UAV now depends not only on the maximum linear velocity of the UAV, v^d_{max}, and the number of nodes N_q (i.e., length of discretized firespot list $\{Q\}^{N_q}$), but it also depends on the velocity of the propagating firespots. We derive the upper-bound on the time it takes a UAV to revisit each firespot location (i.e., each node of the search graph) for Case 2 as introduced in Proposition 2. In Equation 5.9, $\left.\widehat{\dot{q}^{\xi}_{t}}\right|_{\alpha}$ is the estimated linear velocity of the spot q_t in the $\xi \in \{x, y\}$ direction evaluated at confidence interval defined by α.

Proposition 2 *The upper-bound time for a UAV to complete a monitoring tour, $T^{(C2)}_{UB}$, in Case 2, the moving target points, can be calculated as shown in Equation 5.9.*

$$T^{(C2)}_{UB} = \frac{8\zeta^{\alpha}\left(N_q - 1\right)\Delta\mathscr{T}^{MST}}{v^d_{max}\left(1 - 4\zeta^{\alpha}\left(N_q - 1\right)\right)} \tag{5.9}$$

In Equation 5.9, ζ^{α} is defined as in Equation 5.10 and we assume a worst-case velocity for all firespots propagating at the fastest fire's rate of speed in the x- and y-directions, as represented by ζ^{α}. Tuning α enables control of the degree of confidence in our system at the cost of making the UAV coordination problem more difficult. We emphasis that our independence and uniformity of fastest and universal velocity assumptions for firespots pose a worst-case scenario assumption. In other words, the actual service time, T^*, will not become greater than our upper-bound time, T_{UB}, if these assumptions are relaxed.

$$\zeta^{\alpha} = \underset{q,q'}{\arg\max} \sqrt{\left(\widehat{\dot{q}_t^x}\Big|_{\alpha}\right)^2 + \left(\widehat{\dot{q}_t^y}\Big|_{\alpha}\right)^2} \tag{5.10}$$

Proof 2 To arrive at the bound in Equation 5.9, we start by considering the temporal cost of traveling to each firespot under the stationary case, $T_{UB}^{(C1)}$. Moving firespots (even with approximately similar linear velocities), will lead to graph nodes moving possibly in different directions and therefore, the edges of the initial MST graph will shrink or expand. As demonstrated in the middle plot in Figure 5.2, although the blue arrows, representing velocity and direction of each moving node, have the same lengths (i.e., equal velocities), the next-step tour of the UAV (the dashed black line) is longer than the current tour (solid black line). In the worst case of firespots moving in opposite directions, each edge of the search graph expands according to two times the velocity of the fastest firespot, $2\zeta^{\alpha}$. Also, if the graph has N_q nodes for the UAV to consider, this yields $(N_q - 1)$ MST edges and $2(N_q - 1)$ Hamiltonian paths in-between the nodes. We note that the total expansion (shrinkage) of the graph is a function of the time the fire is able to expand (shrink), which is a self-referencing relation, as shown in Equation 5.11. Next, by factoring in the universal velocity of the nodes, ζ^{α}, and replacing $T_{UB}^{(C1)}$ in Equation 5.11 by its value in Equation 5.8 and then solving for $T_{UB}^{(C2)}$, we arrive at Equation 5.9, as demonstrated below.

$$T_{UB}^{(C2)} = 4\zeta^{\alpha}\left(N_q - 1\right)\left(T_{UB}^{(C1)} + T_{UB}^{(C2)}\right) \tag{5.11}$$

$$T_{UB}^{(C2)}\left(1 - 4\zeta^{\alpha}\left(N_q - 1\right)\right) = 4\zeta^{\alpha}\left(N_q - 1\right)T_{UB}^{(C1)} \tag{5.12}$$

$$T_{UB}^{(C2)}\left(1 - 4\zeta^{\alpha}\left(N_q - 1\right)\right) = \frac{8\zeta^{\alpha}\left(N_q - 1\right)}{v_{max}^{d}}\Delta\mathscr{T}^{\text{MST}} \tag{5.13}$$

$$T_{UB}^{(C2)} = \frac{8\zeta^{\alpha}\left(N_q - 1\right)\Delta\mathscr{T}^{\text{MST}}}{v_{max}^{d}\left(1 - 4\zeta^{\alpha}\left(N_q - 1\right)\right)} \tag{5.14}$$

We note that in Case 2, the UAV's FOV becomes irrelevant as the tour is based on UAVs' inference of the future distribution of the fire, and the location is not growing so that it would escape the FOV after T_{UB} steps into the future. Moreover, in the aerial firefront tracking application, when a fire moves to an area that is already burnt or otherwise is observed to dissipate, the fire is pruned from consideration, and the UAV path is updated as there is no fuel for the fire to actually move there.

Case 3: Moving-Spreading Target Points – In the third scenario that we consider, the firespots move and grow quickly, as in a propagating and spreading wildfire. In this case, Single nodes of fire expand over time and escape the UAV's FOV. As such, we must consider the time it takes for a spawning point, q_t, to grow large enough to escape a UAV's FOV, given the maximum leaner velocity, v^d_{max} and current FOV width, w_t, of the UAV. The FOV width for a UAV is directly related to the UAV's current altitude, p^z_t, and the camera half-angle, ϕ, and can be calculated as $w_t = 2p^z_t \tan\phi$. We derive the upper bound for UAV's traversal time allowed to maintain the track quality of each firespot in Case 3 as introduced in Proposition 3. Equation 5.15.

Proposition 3 *The upper-bound time for a UAV to complete a monitoring tour, $T^{(C3)}_{UB}$, in Case 3, the moving-spreading target points, can be calculated as shown in Equation 5.15. We refer to the term, $\mathscr{V}$, in Equation 5.15 as the velocity ratio constant, which captures the ratio between the velocities of the firespots and the UAV and is computed $\mathscr{V} = \frac{2\zeta^\alpha N_q}{v^d_{max}}$.*

$$T^{(C3)}_{UB} = \frac{\mathscr{V} + sqrt\left((1-\mathscr{V})^2 - \frac{64\mathscr{V}(N_q-1)\Delta\mathscr{T}^{MST}(\zeta^\alpha)^2}{v^d_{max}w_t\left(1-4\zeta^\alpha\left(N_q-1\right)\right)}\right) - 1}{4\mathscr{V}\zeta^\alpha(w_t)^{-1}} \tag{5.15}$$

Proof 3 To arrive at the bound for Case 3 as in Equation 5.15, we start by quantifying the maximum planar width and length of the enlarging area. For a specific firespot, q_t, the time-varying planar width and length of the enlarging area along X and Y axes can be computed as $\mathscr{W}(t) \leq 2\,\widehat{\dot{q}^x_t}\Big|_\alpha T^{(C3)}_{UB}$ and $\mathscr{L}(t) \leq 2\,\widehat{\dot{q}^y_t}\Big|_\alpha T^{(C3)}_{UB}$ at the α confidence level, as firespots are now allowed to spread for a maximum of $T^{(C3)}_{UB}$ units of time before the

UAV revisits them. Assuming a vertical scanning pattern, we round up the maximum possible $\mathscr{W}(t)$ and thus, the total number of passes the UAV would need to take from left to right can be calculated as $n(t) \leq \left\lceil \frac{2\widehat{\dot{q}_t^x}\Big|_\alpha T_{UB}^{(C3)}}{w_t} \right\rceil$, and the total distance traversed for each pass is obtained by $d(t) = 2\,\widehat{\dot{q}_t^y}\Big|_\alpha T_{UB}^{(C3)}$. Accordingly, the total pass traversed is given by $d^{tot}(t) = n(t)d(t)$. Therefore, the time it takes one firespot to escape the FOV of the UAV can be calculated as in Equation 5.16.

$$\tau_q(t) = \frac{d^{tot}(t)}{v_{max}^d} = \frac{n(t)d(t)}{v_{max}^d} = \frac{\left\lceil \frac{2\widehat{\dot{q}_t^x}\Big|_\alpha T_{UB}^{(C3)}}{w_t} \right\rceil 2\,\widehat{\dot{q}_t^y}\Big|_\alpha T_{UB}^{(C3)}}{v_{max}^d} \tag{5.16}$$

In order to account for all of the firespots, we compute the summation over all τ_q in Equation 5.16. However, the center of each spreading fire point also moves; thus, $T_{UB}^{(\mathrm{C2})}$ from Equation 5.9 must also be added to this summation, as shown in Equation 5.17.

$$T_{UB}^{(C3)} = T_{UB}^{(\mathrm{C2})} + \sum_{q\in\{Q\}} \frac{2}{v_{max}^d}\,\widehat{\dot{q}_t^y}\Big|_\alpha T_{UB} \left\lceil \frac{2\,\widehat{\dot{q}_t^x}\Big|_\alpha T_{UB}^{(C3)}}{w_t} \right\rceil \tag{5.17}$$

To solve Equation 5.17 for $T_{UB}^{(C3)}$ and find the final upper-bound, we make two simplifying assumptions. First, we remove the ceiling operator and add one to the term inside the operator to achieve continuity, as shown in Equation 5.18.

$$\left\lceil \frac{2\,\widehat{\dot{q}_t^x}\Big|_\alpha T_{UB}^{(C3)}}{w_t} \right\rceil \leq \frac{2\,\widehat{\dot{q}_t^x}\Big|_\alpha T_{UB}^{(C3)}}{w_t} + 1 \tag{5.18}$$

Second, we adopt a similar approach to Case 2 by assuming the area required to cover to account for the growth of each fire location is upper-bounded by N_q times the area of growth for a hypothetical fire growing quickest (Equation 5.19).

$$\sum_{q} \widehat{\dot{q}_t^y}\Big|_{\alpha} T_{UB} \left\lceil \frac{2\, \widehat{\dot{q}_t^x}\Big|_{\alpha} T_{UB}^{(C3)}}{w_t} \right\rceil \leq N_q \zeta^{\alpha} \left\lceil \frac{2\zeta^{\alpha} T_{UB}^{(C3)}}{w_t} \right\rceil \tag{5.19}$$

With these two conservative assumptions and replacing $T_{UB}^{(\mathrm{C2})}$ from Equation 5.9, the upper-bound service time in Equation 5.17 can be revised as shown below, in Equation 5.20-Equation 5.21.

$$T_{UB}^{(C3)} = T_{UB}^{(\mathrm{C2})} + \frac{2N_q \zeta^{\alpha} T_{UB}^{(C3)}}{v_{max}^{d}} \left[\frac{2\zeta^{\alpha} T_{UB}^{(C3)}}{w_t} + 1 \right] \tag{5.20}$$

$$T_{UB}^{(C3)} - \frac{2N_q \zeta^{\alpha} T_{UB}^{(C3)}}{v_{max}^{d}} \left[\frac{2\zeta^{\alpha} T_{UB}^{(C3)}}{w_t} + 1 \right] = \frac{8\zeta^{\alpha}\left(N_q - 1\right)\Delta\mathscr{T}^{\mathrm{MST}}}{v_{max}^{d}\Big(1 - 4\zeta^{\alpha}\left(N_q - 1\right)\Big)} \tag{5.21}$$

We note that Equation 5.21 is in the form of a general quadratic equations (i.e., $z - az(bz+1) = \delta$) in which $z = T_{UB}^{(C3)}$. This form can be reorganized into $\gamma z^2 - \beta z + \delta = 0$, where $\gamma = ab = \frac{4N_q\left(\zeta^{\alpha}\right)^2}{w_t v_{max}^{d}}$ and, $\beta = 1 - a = 1 - \frac{2N_q\zeta^{\alpha}}{v_{max}^{d}}$ and, $\delta = \frac{8\zeta^{\alpha}\left(N_q-1\right)\Delta\mathscr{T}^{\mathrm{MST}}}{v_{max}^{d}\left(1-4\zeta^{\alpha}\left(N_q-1\right)\right)}$. The upper-bound traversal time, $T_{UB}^{(C3)}$, for Case 3 can be obtained from the general form of solutions to quadratic equations, $T_{UB}^{(C3)} = \frac{-\beta+\sqrt{\beta^2-4\gamma\delta}}{2\gamma}$, in which replacing γ, β and δ results in Equation 5.22. Finally, plugging in the velocity ratio constant, $\mathscr{V}$, into the Equation 5.22 and rearranging the terms obtains $T_{UB}^{(C3)}$ as presented in Equation 5.15.

$$T_{UB}^{(C3)} = \frac{\left(\frac{2N_q\zeta^{\alpha}}{v_{max}^{d}} - 1 + \sqrt{\left(1 - \frac{2N_q\zeta^{\alpha}}{v_{max}^{d}}\right)^2 - \frac{128N_q(N_q-1)\Delta\mathscr{T}^{\mathrm{MST}}\left(\zeta^{\alpha}\right)^3}{w_t(v_{max}^{d})^2\left(1-4\zeta^{\alpha}\left(N_q-1\right)\right)}} \right)}{8(w_t v_{max}^{d})^{-1} N_q \left(\zeta^{\alpha}\right)^2} \tag{5.22}$$

5.1.5 Coordinated Distributed Field Coverage Module (No Specified Areas)

UAV agents in the team which are not assigned to monitor the firefronts within specified (human-defined) areas of priority directly (i.e., unallocated UAVs), can be used to explore the rest of the wildfire. See Figure 3.1b as an example illustration. When we have access to additional UAV resources that may not be needed for the performance-guaranteed coverage of the prioritized areas, we can deploy such UAVs (i.e., unallocated UAVs) to monitor the rest of the wildfire areas. To this end, in this section we propose a coordinated, distributed field coverage framework for multiple UAVs to collectively cover and monitor the rest of wildfire areas, other than areas specified by human-teams. We show that how as a corollary of deriving our analytical upper-bound service times, T_{UB}, these temporal values can be leveraged to design a simple yet effective coverage strategy.

To design our proposed coordinated, collaborative field coverage algorithm we focus on settings with *Centralized Planning and Distributed Execution* (CPDE) paradigm. In other words, communication between UAV agents is not restricted during the planning phase which is done by a centralized computer; however, during execution of the assigned plans, each UAV only performs locally and can communicate with its neighboring UAVs. We note that the CPDE paradigm is a standard and widely-used setting for multi-agent planning [206, 66, 207]. The steps to our distributed coverage module for non-prioritized areas are summarized in Algorithm algorithm 3 and are elaborated here.

After detecting the fire map and hotspots, a set of firespots based on the aforementioned Steiner zone variable neighborhood search method are generated (see paragraph 5.1.4). The set of nodes are partitioned (i.e., K-means clustering) according to the number of available (i.e., unallocated) UAVs. Each UAV is assigned to one partition by solving a simple constraint satisfaction problem (CSP) with UAVs as variables, partitions as domains, and distance to the partition centroid as constraints (Line 12 in Algorithm algorithm 3). Note that the partitioning step and solving the CSP problem is done centrally, during the centralized planning phase.

Algorithm 3: Stages of the proposed distributed field coverage module when no area is prioritized for surveillance.

```
input  : Obtain the fire-map ${Q_t}^{N_q}$, list of all UAVs ${\text{UAV}}^{N_d}$, set of UAV poses, velocities, etc.
output : Assigned UAV-path pairs $\{\mathscr{A}_t^d\}$
1  objective: Unallocated UAVs, ${\text{UAV}}^{N_d^0}$, to cover unspecified areas, ${Q_t}^{N_q^0}$
2  // main loop //
3  Initialize $t \leftarrow 0$
4  while MissionDuration do
5      Update fire-map and priority areas: $\langle \{Q_t\}^{N_q^1}, \{Q_t\}^{N_q^0} \rangle \leftarrow$ UpdateMap($\{Q_t\}^{N_q}$)
6      Update UAVs status: $\langle \{\text{UAV}\}^{N_d^1}, \{\text{UAV}\}^{N_d^0} \rangle \leftarrow$ UpdateDroneStats($\{UAV\}^{N_d}$)
7      CoordinatedCoverage($\{Q_t\}^{N_q^0}, \{\text{UAV}\}^{N_d^0}, t$) // pass in the time $t$
8      Update time: $t \leftarrow t+1$
9  end
10 // inner functions //
11 def CoordinatedCoverage($\{Q_t\}^{N_q^0}, \{\text{UAV}\}^{N_d^0}, t$):
12     $\langle \{T_{UB_{i1}}^{g'}, .., T_{UB_i}^{g'}\}, \{\mathscr{P}_{i1}', .., \mathscr{P}_i'\} \rangle \leftarrow$ CSP(TUB(Cluster($\{Q_t\}$, len($\{UAV\}^{N_d^0}$))))
13     if $\left(t \geq \min(T_{UB}^i)\right)$ || $\left(i \in \{\text{UAV}\}^{N_d^0}\right.$ is deployed for human support$\left.\right)$
14         $\langle \{\mathscr{P}_{i1}', \cdots, \mathscr{P}_{i_{new}}'\} \rangle \leftarrow$ Cluster($\{Q_t\}$, len($\{UAV\}_{new}^{N_d^0}$)) // re-clustering
15         Repeat line 12 for the new assignments
```

After assigning UAVs to a partition, each UAV agent starts to execute the following tasks distributedly (Line 12 in Algorithm algorithm 3). First, an optimal path for coverage and tracking is found by applying the k-opt algorithm. The upper-bound time, T_{UB}, is then calculated for a firespot in the center of a UAV's FOV. T_{UB} provides the UAV with an estimate of how long it will take a fire to escape its FOV as determined by fire propagation velocity. When a route is identified in this way, UAVs can apply this reasoning for the next T_{UB} time-steps before recalculating a new path (Line 13 in Algorithm algorithm 3). After T_{UB} has passed, the partitions are revised, and an optimal path is recalculated, since fire locations have likely changed significantly during this time (Lines 14-15 in Algorithm algorithm 3). Once a UAV is recruited for help to monitor firefronts within human-specified areas, one of the unallocated UAVs is dispatched and the process is repeated from the central planning phase. In scenarios where the centralization of the planning step does not impose an issue, leveraging this method provides a simple, low-complexity approach that plans for multiple UAVs at the high-level (without considering the low-level control inputs) to efficiently cover a dynamic field.

We emphasize that the relation between our coordinated, distributed field coverage algorithm described here and the coordinated planning framework (described in subsection 5.1.4) is that the latter coordinates UAV agents to efficiently track the firefronts within specified disjoint areas of fire (i.e., areas prioritized by human firefighters) by using as few UAVs as possible, while guaranteeing the performance. The algorithm described in this section utilizes the remaining UAVs in the team (if any) to collectively surveil the rest of the wildfire area, if needed. In general, this module may not be needed, or it can be replaced with any other surveillance methods depending on the problem setting (e.g., if centralized planning is possible or if it needs to be fully distributed). For instance, if the centralized planning is not feasible in an application, the coverage method in this section can be replaced with the fully decentralized controller in [5], or any other similar methods from the literature.

5.1.6 Empirical Evaluation

In this section, we empirically evaluate both our multi-UAV coordination and planning framework (subsection 5.1.4) as well as the coordinated, distributed field coverage module introduced in subsection 5.1.5 in an aerial firefront tracking and wildfire area monitoring case-study. In our experiments, robots in a team of homogeneous, autonomous UAVs (e.g., omni-directional multi-rotor aircrafts such as quadcopters) are tasked to coordinate together to: (1) track and monitor the firefront within multiple specified human-defined vicinities of priority and provide real-time fire states and tracking information and (2) cooperatively cover and surveil the entire wildfire area. For the field coverage we test the performance of our approach in comparison with two state-of-the-art model-based and reinforcement learning benchmarks (see subsubsection 5.1.6). Moreover, we demonstrate the feasibility of our framework through implementation on physical robots in a multi-robot testbed (subsection 5.1.7).

Baselines

The first benchmark to which we compare is a recent distributed control framework for dynamic wildfire tracking as proposed by [36]. The distributed control framework includes two controller modules in which one is responsible for field coverage and the other for path planning (i.e., in-flight collision avoidance, maintaining a safe altitude, and moving towards new desired poses). The two controllers are defined as the negative gradients (gradient descent) of objective functions to maximize the area-pixel density of the UAV's fire observations and to maintain safe flight parameters with potential field-based criteria, respectively.

Second, we compare our approach to a reinforcement learning (RL) benchmark method proposed in [119]. Each UAV is controlled by an independent agent, and all agent policies are identical (scalable RL algorithm). The policy network architecture consists of three fully connected layers with ReLU activations, following prior work [119]. The agent receives as input an uncertainty map over the wildfire, as described in paragraph 5.1.4, and outputs a direction to move for the next time-step. The reward function is adapted from prior work, though elements that involve collision are removed, as we assume UAVs can occupy the same space. We follow hyperparameter settings given in prior work, and perform a sweep over hidden-layer dimensions, as they are not explicitly defined in prior work.

Simulation Environment and Results

Evaluating the Algorithms To evaluate the efficacy of our proposed multi-UAV planning framework for firefront tracking, we performed a comprehensive simulation to determine the number of UAVs needed to satisfy the uncertainty residual ratio (given in Equation 5.6) for a range of numbers of disjoint areas specified for firefront tracking. We performed the evaluation for all three mentioned wildfire scenarios for ten trials and calculated the mean and standard error (SE) for each. The objective was to determine the number of required UAVs to guarantee the performance in each case. The results for this simulations

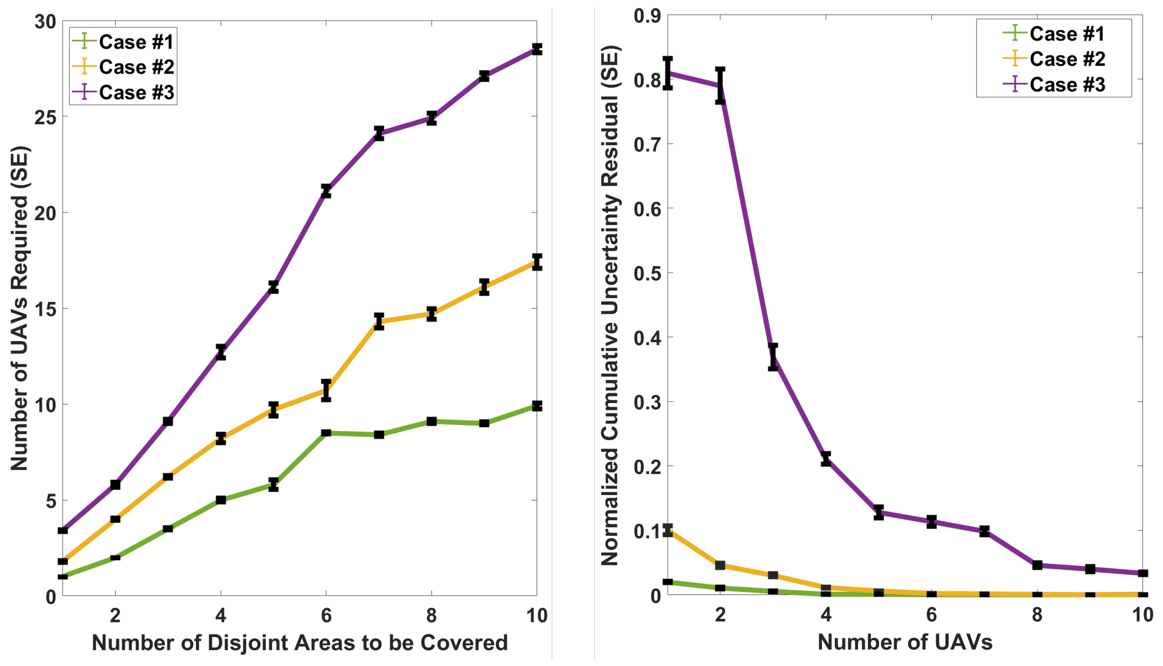


Figure 5.3: The left-side figure depicts a quantitative evaluation of our analytical URR bound and shows that increasing the number of disjoint areas to be monitored resulted in a rise in the number of UAVs required to guarantee the performance. This also held as the wildfire propagation scenario changed and a fire propagated more aggressively. The right-side figure demonstrates the efficacy of the coordinated, distributed coverage algorithm, in which less cumulative uncertainty residuals (Y-axis) is better. This figure shows that UAVs easily covered the fire map in the case of stationary wildfire, while in the case of a fast-growing fire, more UAVs were required to cover the fire map efficiently.

are presented in Figure 5.3 (left-side figure).

For the aforementioned evaluations, we simulated one to ten (i.e., $N_h \in \{1,2,..,10\}$) distant areas of fire, assuming each corresponds to one prioritized area. Each area included 20-30 (i.e., random integer in the range $[20,30]$) randomly placed firespots. Areas were randomly initialized within a 500-by-500 terrain. A UAV team was randomly positioned within a distant location at a corner of the terrain. A total number of 30 UAVs were assumed as the goal was to determine the number of required UAVs to guarantee the quality of service in each case. We chose the fire propagation velocity to be 0, 0.5, and 1 for wildfire Cases 1, 2, and 3, respectively. Moreover, we chose the spawning rate of the fire for Case 3 to be at most three (i.e., each fire can produce up to three more fires). Additionally, the team of UAVs were homogeneous in their dynamics and motion characteristics where the maximum linear velocity, v^d_{max}, of the UAVs was set to 500 for all UAVs.

To further investigate the UAV-team behavior during our multi-UAV planning framework in the above experiment, we demonstrated the computed T_{UB} times by the UAVs in Figure 5.4. Figure 5.4 shows the computed upper-bound times, T_{UB}, that it takes UAVs to complete a determined tour with respect to the three wildfire scenarios. The plots in Figure 5.4 are averaged across UAVs and across ten separate experiment trials. As shown, the computed upper-bound service times initially start with a high value, relative to the wildfire scenario since, at first, only one UAV is assigned to monitor and track the entire firefront points. As the time proceeds and the algorithm determines that more UAVs are required for a guaranteed tracking of the firespots, the computed T_{UB} times become smaller until the values converge to a reasonable time respective to the fire scenario.

Additionally, to demonstrate that our derived upper-bound times, T_{UB}, are tight with respect to the actual maximum time it takes UAVs to complete a tour, T^*, we computed the $\frac{T_{UB}}{T^*}$ ratio for all fire scenarios in an experiment. This empirically-evaluated ratio evaluates the tightness of the derived upper-bound times where for a tight bound, the value of this ratio must remain close to one. Figure 5.5a shows the mean value of this ratio ($\pm$ standard error), computed over 5000 trials for each fire scenario. As shown, the value of the ratio slightly increases as the fire scenario gets more intense. The upper-bounds for Case 1 (i.e., Equation 5.8), Case 2 (i.e., Equation 5.9), and Case 3, (i.e., Equation 5.15) are 1.1$\times$, 1.18$\times$, and 1.48$\times$ greater than the actual maximum times, T^*, for the respective cases. Results show that our upper-bounds become more conservative for more aggressive fire scenarios. We note that for this experiment, the environment parameters are as described above, in paragraph 5.1.6, and the actual maximum times, T^*, were computed directly in simulation and without considering the MST paths designed in our algorithm.

We also evaluated our coordinated, distributed field coverage algorithm within a similar framework by calculating the cumulative uncertainty residual while tasking an increasing number of UAVs to surveil a large propagating wildfire map (i.e., no specified areas). The

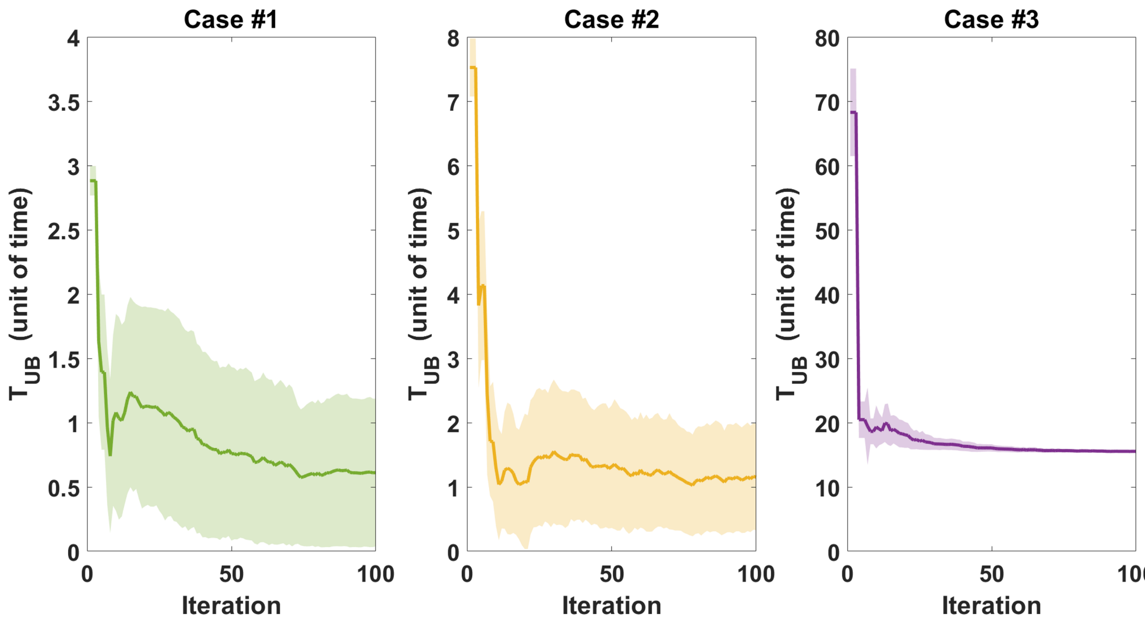


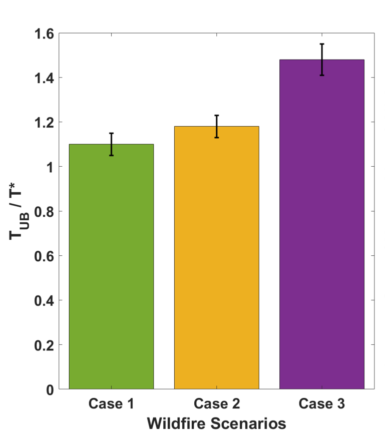
Figure 5.4: The computed upper-bound times, T_{UB}, it takes a UAV to complete a determined tour over the course of a simulation with respect to the three wildfire scenarios. As shown, the computed upper-bound service times initially start with a high value, relative to the wildfire scenario, and gradually decrease until converging to a reasonable value, as the algorithm determines that more UAVs are required for a guaranteed tracking of the firespots. Plots are averaged across UAVs and across ten separate experiment trials.

uncertainty residual at each timestep was calculated by inspecting the ground-truth fire map for firespots that were not covered by any UAVs, and the respective cumulative error residual obtained from AEKF was calculated. The uncertainty residual measure was an indicator of how successful the team of UAVs were in cooperatively covering all firespots and increased by the number of nodes not covered by any UAV. Figure 5.3 (right-side) shows the results for the evaluation of our coordinated, distributed coverage algorithm. All wildfire environment and UAV characteristics and parameters for experiments in Figure 5.3 (right-side) were similar to the description presented above and once again, the experiments were conducted for all three aforementioned wildfire scenarios.

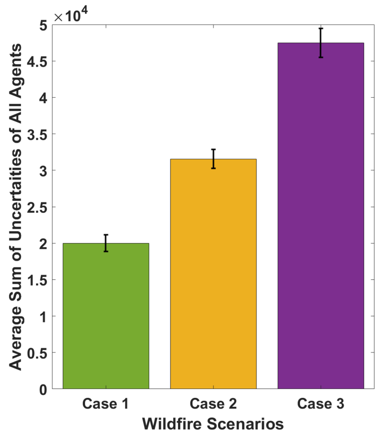
We further investigated the efficacy of our algorithm by: (1) assessing the boundedness of the UAVs' measurement residuals through cumulative uncertainties and (2) evaluating the performance in an evolving wildfire scenario. For these experiments, we initialized two distinct fire areas in a large 500$\times$500 terrain, each with around 30 initial firespots.

Figure 5.5b shows the sum of all agents' uncertainties for all measurements, averaged over time and across ten trials. As shown, the measurement uncertain residuals are bounded while the values increase as the wildfire scenario becomes more aggressive. Figure 5.5d illustrates the combined uncertainty maps in the 500×500 terrain for each of the three wildfire scenarios by summing all the measurement residuals of all agents for all firespots and averaging over time. Figure 5.5c compares the measurement uncertainty residuals for our coordinated field coverage approach and [36] in an evolving fire scenario. For this experiment, firespots start in Case 1 (i.e., stationary) and evolve into Case 2 and 3 at $t = 50$ and $t = 150$, respectively for a total of $t = 350$ time-steps. A total of five UAVs initiated at a distant location are tasked to cover and track the wildfire area at all times. As shown in Figure 5.5c, our method outperforms the baseline in this challenging scenario by accumulating $5.8\times$ less uncertainty residual than the baseline over the simulation time and on average per time-step.

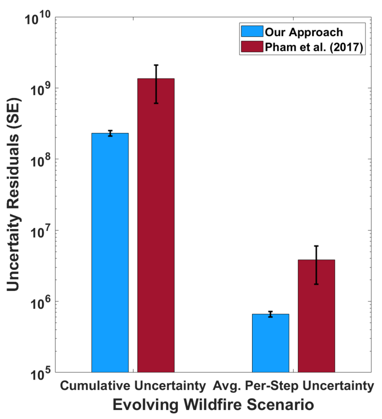
Baseline Comparison Furthermore, we also evaluated our method by assessing its performance in comparison with two state-of-the-art benchmark studies ([36] and [119]) for wildfire coverage. In our simulations, we tested all three algorithms in two different fire environments: (1) our fire environment where one area of fire including 40 randomly-placed firespots was initialized and a total of five randomly positioned UAVs within a distant location were assigned to cover the fire area within a 500-by-500 terrain. (2) the fire environment utilized in [119] where a total of 16 firespots all within a 4-by-4 square in the center of a 50-by-50 terrain were initialized and a total of 10 UAVs positioned around this initial square were assigned to cover the fire area. For both environments, UAVs were required to cover the entire fire map as much as possible, while maintaining an altitude that was both safe and conducive to high-quality imaging. Moreover, the fire model parameters, R_t, U_t and θ_t, were initialized as in [36] for comparison. The results for these evaluations are presented in Figure 5.6 in which top and bottom rows correspond to (1) our fire environment and (2) that



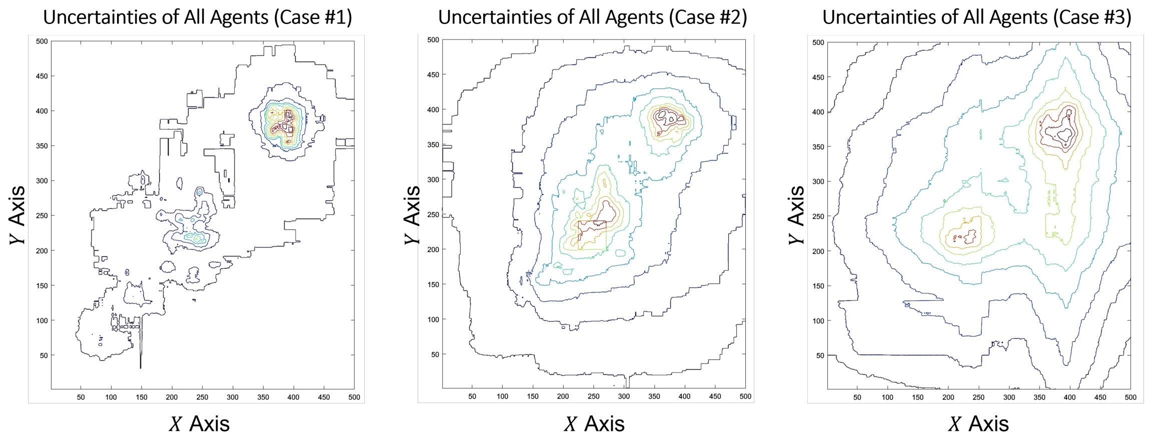
(a) $\frac{T_{UB}}{T^*}$ ratio for all fire scenarios.



(b) Average sum of all uncertainties over time.



(c) The uncertainty residuals in evolving fire scenario.



(d) The combined uncertainty maps generated by all UAV agents for all firespots, averaged over time.

Figure 5.5: Figure 5.5a shows the $\frac{T_{UB}}{T^*}$ ratio for all fire scenarios computed over 5000 trials for each case. This ratio evaluates the sanity of the derived upper-bound times, T_{UB}, and the actual maximum time it takes UAVs to complete a tour, T^*. As shown, for a tight upper-bound, the value of this ratio must remain close to one. Figure 5.5b shows the sum of all UAVs' uncertainties for all of the measurements, averaged over time. As shown, the measurement uncertain residuals are bounded. Figure 5.5c compares the measurement uncertainty residuals for our approach and [36] in the evolving fire scenario (i.e., fire starts in Case 1 and evolves into Case 2 and 3 at certain time-steps). As shown, our method outperforms the baseline in this more challenging scenario. Figure 5.5d shows the combined uncertainty maps generated by summing all measurement residuals of all UAVs for all firespots and averaging over time.

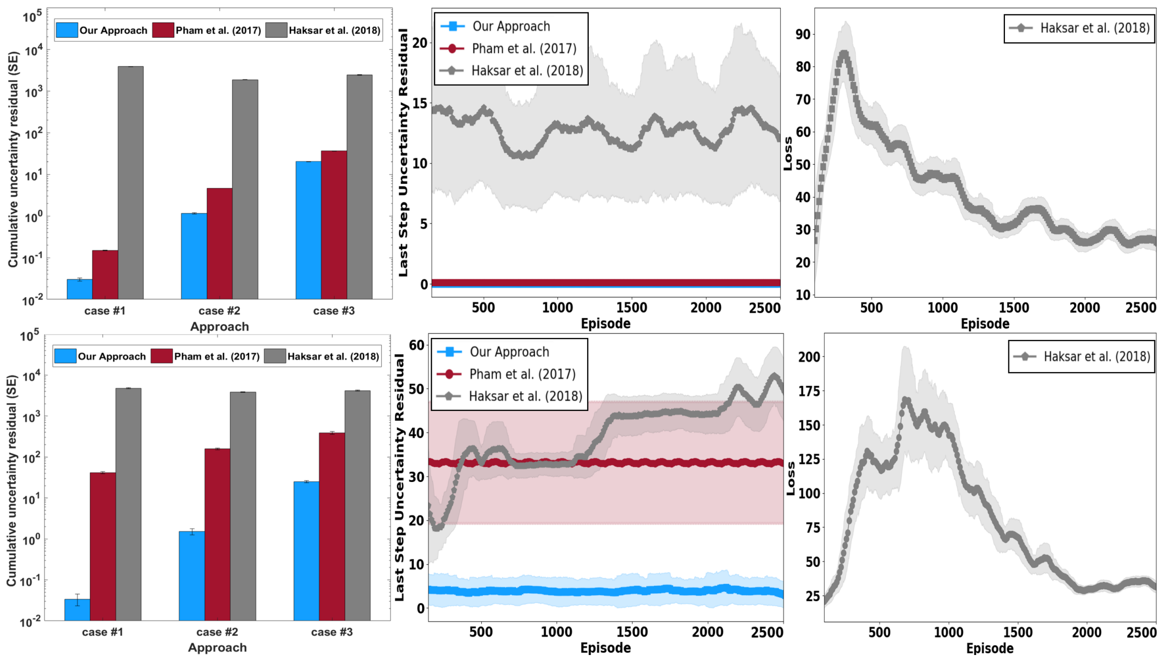


Figure 5.6: This figure presents a comparison of our predictive coordinated controller to prior works for both our fire environment (bottom row) and that of [119] (top row). First column figures show the cumulative uncertainty residual during 100 simulation time steps (10 trials). Middle column figures show the uncertainty residual at the end of last simulation step over the course of 2500 episodes of 100 time-step simulation. The third column figures show the RL agent's loss over the course of training.

of [119], respectively.

Discussion

We investigated the sensitivity of our algorithm to the fire scenario, number of separate areas to be monitored, and the number of UAVs available, as shown in Figure 5.3. The left-hand figure shows that increasing the number of disjoint areas to be monitored resulted in a rise in the number of UAVs required to guarantee the performance. This also held as the wildfire propagation scenario changed and a fire propagated more aggressively. As represented in Figure 5.3 (right-side figure), the UAV team was able to easily cover the fire map in the case of stationary wildfire, while in the case of a fast-growing fire, more UAVs were required to cover the fire map efficiently (i.e., with low cumulative measurement uncertainty residual).

As shown in Figure 5.4, the computed upper-bound service times, T_{UB} initially start with a high value and gradually decrease until converging to a reasonable value, as the algorithm determines that more UAVs are required for a guaranteed tracking of the firespots. Note that the initial and the converged values of the T_{UB} times have higher values for wildfire Cases 2 and 3 since the moving and moving-spreading firefronts result in expanded search graphs.

The comparison between our approach and control-theoretic [36] and reinforcement learning-based [119] benchmarks shows that we are able to achieve a $7.5\times$ and $9.0\times$ reduction in error residual for the most challenging cases, as depicted in Figure 5.6. For all cases, our approach achieved significantly lower uncertainty residual and cumulative uncertainty in both fire environments. These results demonstrate that our framework not only provides probabilistic bounds on guaranteeing performance, but also achieves an empirically more-optimal solution for maintaining a tight track on wildfire propagation. Herein, we declare that the main objective for these benchmark comparisons is to evaluate the soundness and feasibility of our coordinated field coverage approach in subsection 5.1.5 as compared to standard state-of-the-art approaches for this purpose and to evaluate whether our *more-informed* approach can fulfill the expectations by producing comparable and/or better results.

Moreover, we note that the reward-function specification problem and consequently scalability issue in RL-based methods are present, as the RL agent fails to achieve good performance even after convergence (third column from left in Figure 5.6). Possible reasons for this include an over- or under-specified reward function from prior work, which emphasized fixed-wing aircraft flight patterns and did not explicitly encourage maximal coverage.

We note that, as discussed in paragraph 5.1.4, an unsatisfied URR bound means that the uncertainty residual may grow with each tour of the UAVs, which is an indicator that the UAVs may not be capable of monitoring the targets without losing the track quality. Therefore, a URR greater than one only means that the tracking quality cannot be guaranteed in the respective application. However, the strictness of this bound and whether a UAV can continue the tracking task with small growth in the uncertainty (e.g., 10% growth) depend on the rate of this growth over time and if such growth is acceptable in the respective application.

Additionally, the URR upper-bound is designed to recruit UAVs into the monitoring team. However, the URR bound also can be readily used for dismissing UAVs in a simple procedure such as: if URR bound is satisfied with n UAVs then check URR with $n-1$ UAVs (graphs and tours must be recalculated with $n-1$) and set $n = n-1$ if URR is still satisfied. Otherwise, n UAVs will remain. Note that, the URR bound is checked by each UAV and for all the firespots in their assigned region. To select a UAV to be dismissed when the URR bound is satisfied with $n-1$ UAVs, we can choose from regions that are close to each other. This can be achieved, for instance, by calculating the relative distances between the region centroids. When a UAV is dismissed in this way, the reclustering process of the regions is performed through a centralized clustering step such as applying a k-means with $n-1$ desired clusters. We then assign UAVs to their new regions.

5.1.7 Demonstration: Multi-robot Testbed

In order to account for vehicle dynamics and motion constraints, we implemented and tested our coordinated planning and distributed field coverage modules in the Robotarium multi-agent robotic platform [151], at the Georgia Institute of Technology. The simulated growing wildfires using the introduced FARSITE [122] model are projected on the arena as the regions to be covered by the robot team. The fire simulation parameter setup for our experiments are similar to the empirical evaluation case. In Figure 5.17 and Figure 5.8, the sub-figures (1)-(4) illustrate initial to final robots standings in our experiments for (1) coordinated planning algorithm for tracking firefronts within specified separate areas and (2) the coordinated, distributed field coverage method, respectively. The boxes around each robot shows their respective altitude-dependent FOV. All experiments are repeated for the three wildfire cases and video recordings of these experiments can be found on

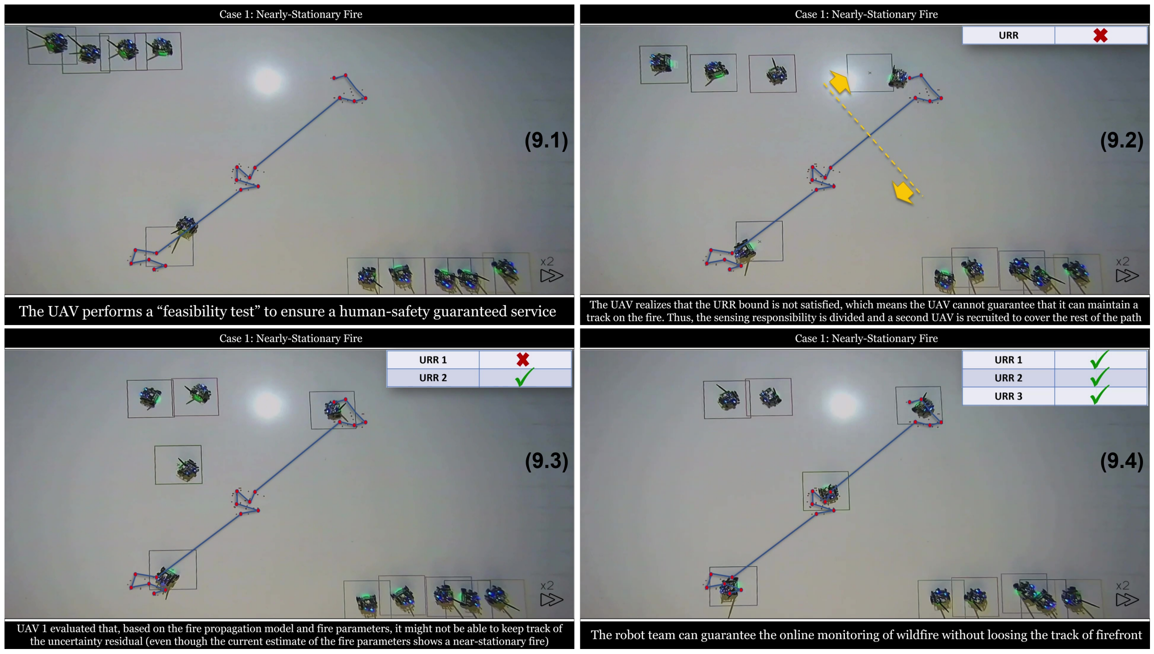

Case 1: Nearly-Stationary Fire
(9.1)
x2
The UAV performs a "feasibility test" to ensure a human-safety guaranteed service
Case 1: Nearly-Stationary Fire
URR ✗
(9.2)
x2
The UAV realizes that the URR bound is not satisfied, which means the UAV cannot guarantee that it can maintain a track on the fire. Thus, the sensing responsibility is divided and a second UAV is recruited to cover the rest of the path
Case 1: Nearly-Stationary Fire
URR 1 ✗
URR 2 ✓
(9.3)
x2
UAV 1 evaluated that, based on the fire propagation model and fire parameters, it might not be able to keep track of the uncertainty residual (even though the current estimate of the fire parameters shows a near-stationary fire)
Case 1: Nearly-Stationary Fire
URR 1 ✓
URR 2 ✓
URR 3 ✓
(9.4)
x2
The robot team can guarantee the online monitoring of wildfire without loosing the track of firefront

Figure 5.7: This figure presents example demonstrations of the proposed coordinated planning algorithm for human safety, implemented on physical robots. In Figure (9.4) all URRs are satisfied and the algorithm determines that a team of three robots can collectively provide the estimated information for the three areas without losing the track of any spot. Link to video: `https://youtu.be/zTR07cKlwRw`.

`https://youtu.be/zTR07cKlwRw`.

For the coordinated planning algorithm with URR condition for performance guarantee, we specified ten robots and tested the feasibility of the algorithm for all three wildfire cases. In the first case, only three robots were needed to guarantee the quality of service at all times, while this number was six and nine for the second and third scenarios, respectively. Figure 5.17 presents example demonstrations of these experiments. Figure 9.1 shows the beginning of the experiment where a single robot is required to track and monitor three distant areas of fire (i.e., simulated firespots projected on the arena). The robot path is also shown. In Figure 9.2 the robot determines it cannot guarantee to provide real-time information about all three areas since the URR (shown on top left) is not satisfied and thus, the map is partitioned and another robot is summoned. The same process is repeated in Figures 9.3 – 9.4 until all URRs are satisfied and the three robots can collectively provide the estimated information for the three areas without losing the track of any spot. Video

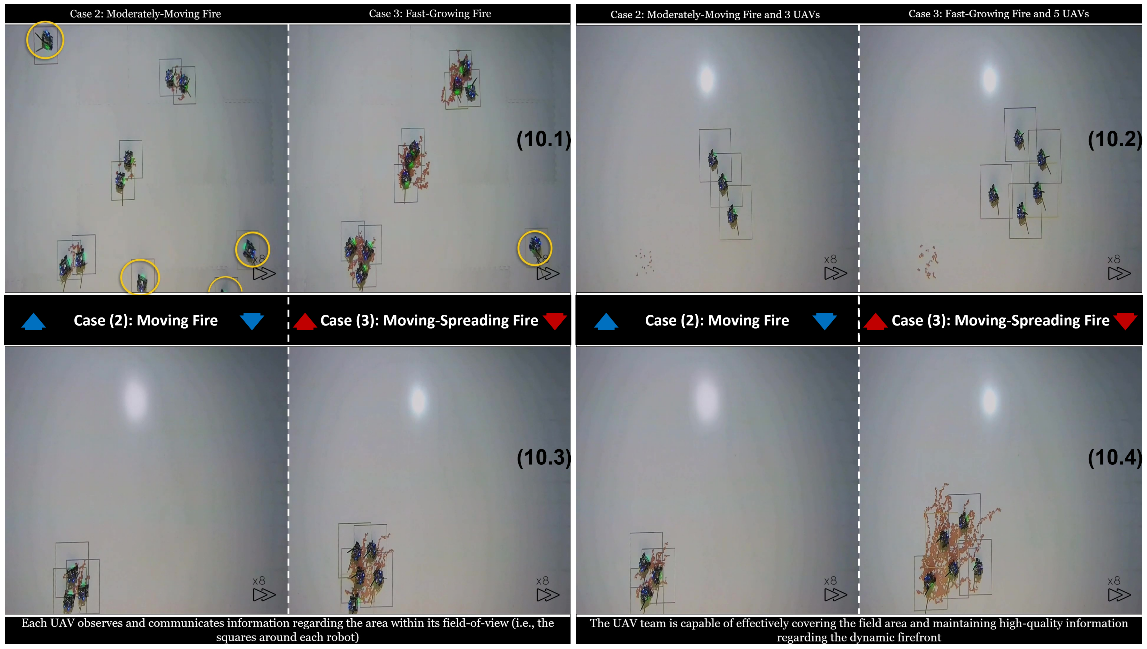

Case 2: Moderately-Moving Fire
Case 3: Fast-Growing Fire
Case 2: Moderately-Moving Fire and 3 UAVs
Case 3: Fast-Growing Fire and 5 UAVs
(10.1)
(10.2)
Case (2): Moving Fire
Case (3): Moving-Spreading Fire
Case (2): Moving Fire
Case (3): Moving-Spreading Fire
(10.3)
(10.4)
x8
Each UAV observes and communicates information regarding the area within its field-of-view (i.e., the squares around each robot)
The UAV team is capable of effectively covering the field area and maintaining high-quality information regarding the dynamic firefront

Figure 5.8: This figure presents four sample demonstrations of our experiments for the proposed coordinated field coverage algorithm, implemented on physical robots. Each sub-figure 10.1 – 10.4 includes two images: (1) moderate moving fire (left-side) and (2) fast moving-spreading fire (right-side). Figure 10.1 represents the initial step in the algorithm in which *unallocated* robots are selected. Figure 10.4 shows that the robot team can successfully cover both wildfire cases and monitor the dynamic spots, after ten minutes of running the algorithm. Link to video: `https://youtu.be/zTR07cKlwRw`

recordings of these experiments for all three wildfire scenarios can be found on the *first* part of the provided supplementary video.

Next, we tested the coverage performance using five and three robots, again for all three wildfire scenarios. Figure 5.8 presents four sample demonstrations of our experiments for the proposed coordinated field coverage algorithm, implemented on physical robots. Each sub-figure 10.1 – 10.4 includes two images: (1) moderate moving fire (left-side) and (2) fast moving-spreading fire (right-side). Figure 10.1 represents the initial step in the algorithm in which *unallocated* robots are selected. Figures 10.2 and 10.3 show the team of robots swarming towards the wildfire area and beginning the surveillance. Figure 10.4 shows that the robot team can successfully cover both wildfire cases and monitor the dynamic spots, after ten minutes of running the algorithm. Video recordings of these experiments can be found on the *second* part of the provided supplementary video.

5.1.8 Limitations and Future Work

Our coordinated planning algorithm is model-based. In our framework, UAVs quantify their estimation uncertainty by explicitly inserting the inferred model parameters from the environment into the model and following a nonlinear uncertainty propagation law. According to our experiments, a mismatch between the actual model used by the algorithm and the ground truth model can lead to significantly increasing the actual measurement uncertainty residuals and consequently an unreliable coverage plan. While this poses a limitation on the framework, in such cases, the best practice is to use adaptive estimation methods such as Adaptive EKF (AEKF) where the covariances can change adaptively. Nevertheless, the results of such methods can still be unreliable.

We performed an experiment to evaluate the performance of the AEKF in our framework for cases when there exists a mismatch between the process model and the actual fire propagation model. To impose a model mismatch, we altered the $LB(U_t)$ equation in the actual FARSITE model (introduced in section 3.2) to be $LB_{new} = \mathscr{X}_1.LB(U_t) + \mathscr{X}_2$, where $\mathscr{X}_1 \sim \mathscr{N}(10,3)$ and $\mathscr{X}_2 \sim \mathscr{N}(0,3)$. This alteration would also alter the values of $GB(U_t)$ and $C(R_t,U_t)$. However, we did not revise the derivatives in the process Jacobian matrix, F_t, to account for this alteration. The rest of the environment parameters were exactly the same as in paragraph 5.1.6. We repeated the simulation for ten separate trials and recorded the cumulative uncertainties over each run. As a result, our experiments demonstrate that the described mismatch between the actual FARSITE model and the process model in the adopted AEKF, on average, caused 1.48 ± 0.4 times higher cumulative uncertainties across wildfire scenarios, as compared to the case without a model mismatch. This increase in uncertainty residual demonstrates that even an adaptive estimation method such as AEKF cannot completely compensate for a model mismatch.

We discuss that in our approach both UAVs or a ground *control room* can be in charge of running the algorithm. By assuming local communication between neighboring UAVs, without losing generality, we can presume either case to be practical, assuming required hardware and computational resources are available. By assuming an on-board computer, each UAV can perform the URR bound check in Equation 5.6 and communicate its local belief of its assigned path to a ground control room, which then recruits additional UAVs to the task. We note that distributed communications could result in bottlenecks in our coordinated field coverage method; however, we rely on the rich literature on algorithms available to manage wireless communication grids to address this problem (see [208, 209]).

In paragraph 5.1.4, we made a worst-case scenario assumption regarding the independence of firespots and uniformity of fastest and universal velocity for all firespots. While these assumptions are made to generalize the applicability of the approach, in future work, the independence assumption can be relaxed for the wildfire monitoring application, given that an accurate model of the correlations between firespots are in hand. Accordingly, the calculated upper-bound service time, T_{UB}, will become smaller and closer to the actual service time, T^*, making the generated monitoring plans more optimal.

5.1.9 Conclusion

We have introduced a novel analytical measurement-residual bound on fire propagation uncertainty, allowing high-quality planning for real-time wildfire monitoring and tracking, while also providing a probabilistic guarantee on the quality of service. Our approach outperformed prior work for distributed control of UAVs for wildfire tracking, as well as a reinforcement learning baseline. Our quantitative evaluations validate the performance of our method accumulating 7.5$\times$ and 9.0$\times$ less error residual than the model based benchmark [36] and the learning-based benchmark [119] respectively when covering a large aggressive wildfire, over a course of 2500 episodes of 100-steps long simulations. Our method also outperformed the best-performing baseline [36] in a challenging evolving fire scenario by accumulating 5.8$\times$ less uncertainty residual than the baseline over the simulation time and on average per time-step method. See subsection 5.1.6 and subsubsection 5.1.6 for the evaluations and a discussion on the presented results. Physical implementation of our framework on real robots in a multi-robot testbed demonstrate and validate the feasibility of our approaches.

We note that in this work we generalize by considering large-scale human-defined areas of priority for monitoring and tracking moving targets (such as firespots) and investigating three different scenarios of stationary targets (e.g., the traveling salesman problem and search-and-rescue), moving targets (e.g., border patrol and the tracking of wildlife poachers) and moving-spreading targets (e.g., wildfire monitoring and oil spill surveillance). We also note that due to the modular design of our framework here, any distributed control algorithm for dynamic field coverage, such as our introduced previous work [5], can replace the proposed coordinated coverage module in subsection 5.1.5 to enable *unallocated* UAVs to monitor the unspecified areas of wildfire.

5.2 A Hierarchical Coordination Framework for Joint Perception-Action Tasks in Composite Robot Teams

In this section, we propose a collaborative planning and control algorithm to enhance cooperation for composite teams of autonomous robots in dynamic environments. Composite robot teams, as introduced in subsection 3.1.2, are groups of agents that perform different tasks according to their respective capabilities in order to accomplish an overarching mission. Examples of such teams include groups of perception agents (can only sense) and action agents (can only manipulate) working together to perform disaster response tasks. Coordinating robots in a composite team is a challenging problem due to the heterogeneity in the robots' characteristics and their tasks. Here, we propose a coordination framework for composite robot teams. The proposed framework consists of two hierarchical modules: (1) a Multi-Agent State-Action-Reward-Time-State-Action (MA-SARTSA) algorithm in Multi-Agent Partially Observable Semi-Markov Decision Process (MA-POSMDP) as the high-level decision-making module to enable perception agents to learn to surveil in an environment with an unknown number of dynamic targets and (2) a low-level coordinated control and planning module that ensures probabilistically-guaranteed support for action agents. Simulation and physical robot implementations of our algorithms on a multi-agent robot testbed demonstrated the efficacy and feasibility of our coordination framework by reducing the overall operation times in a benchmark wildfire-fighting case-study.

5.2.1 Introduction and Motivation

Multi-robot teams are able to execute time-sensitive, complex missions by cooperatively leveraging their unique capabilities and design [52]. Heterogeneity in robots' design characteristics and their roles are introduced to (1) leverage the relative merits of different agents and their capabilities [52] and, (2) deal with the dynamic and unpredictable nature of the real-world for which designing homogeneous, versatile robot teams that can effectively adjust to all circumstances is difficult and costly [210]. This introduction entails the formation of *composite teams* of heterogeneous, co-dependent agents, in which different robots speciate in different parts of a compound or complex task (see Korsah et al. [211] for a complete taxonomy of tasks in multi-robot teams).

An important instance of composite robot-teams is the collaboration between *perception (sensing) agents* and *action (manipulator) agents* [53, 212]. In a perception-action composite robot team, to complete a mission, perception robots are first, tasked to explore an unknown environment to find an initial set of dynamic targets and then, *exploit* those targets. Estimated target-states are required by action agents to perform a specific manipulation on those targets. In an example aerial wildfire fighting task, perception UAVs first explore the environment to find fire spots. Once key fire spots are identified, the perception agent will loiter to gain a higher quality state estimate of the fire targets for an action UAV to be able to efficiently douse each fire, thereby *exploiting* the targets. In this paper, we make the common assumption that perception robots are only capable of sensing while action robots are only capable of manipulating. We note that in dynamic environments, the described exploration versus exploitation trade-off for perception robots needs to be decided in a *restless* manner [213], in which targets' states, such as position and velocity, evolve over time regardless of agents' actions. Applications of such composite robot teams are numerous in surveillance and disaster response [53, 55, 58], search and rescue [214, 215], manufacturing [83], and border patrol [216].

Previous studies tackle various aspects of heterogeneous multi-robot teaming by considering coordination strategies [55, 212, 217, 218], task allocation [56, 219, 33], and path-planning and control [57, 8, 219, 220, 221]. The problem of coordination and collaborative planning between perception and action agents has been of keen interest to the wireless communication research community, referring to the problem as *Wireless Sensor and Actor Networks* [53, 58]. While most of this prior work studies static environments (e.g., [222, 55, 58, 83, 214]), this assumption does not hold true in many environments of interest which are dynamic (e.g., including numerous, moving targets). Such dynamic environments have attributes of both a Partially Observable Markov Decision Process (POMDP) and a Restless Bandit Problem [59]. Unfortunately, traditional Reinforcement Learning (RL) formulations lack the scalability and adaptability with respect to domain shift in order to tackle real-world problems. Moreover, most of the proposed nonlearning-based approaches (e.g., mixed-integer linear/non-linear program such as [212]) fail to handle the large-scale, dynamic, and stochastic nature of these problems.

Efficient planning and coordination of robots with different traits in a composite team while accounting for their collaborative behavior through specific capabilities and limitations is of significant importance [53, 52]. This coordination becomes more challenging when the dynamicity of the environment needs to be taken into account [83].

Contributions

Our work overcomes the limitations of prior work by proposing a novel hierarchical approach (Figure 5.9) to tackle the (1) high-level decision-making and (2) the low-level coordinated control problems for heterogeneous teams of autonomous robots consisting of *perception-only* and *action-only* agents. The high-level decision-making module is centralized and is responsible for effectively balancing the exploration (of the environment) vs. exploitation (of found targets) trade-off for perception agents. The low-level module is a fully distributed coordinated controller that guides the perception agents to generate action-agent-aware, performance-guaranteed trajectories for action agents. To enable our framework, we propose a novel RL algorithm leveraging the State-Action-Reward-State-Action (SARSA) algorithm within a Multi-Agent Partially Observable *Semi*-MDP (MA-POSMDP); our approach enables us to learn a policy for perception agents that addresses the high-level exploration versus exploitation dilemma in an unknown, dynamic environment in support of action agents. We consider an unknown, variable number of moving targets with known motion models to be explored by the perception robots, while also tasking them to exploit the found targets to (1) extract (estimate) the necessary state-information, e.g. dynamics such as instantaneous position and velocity, (2) generate a feasible trajectory for action robots, and finally (3) task action agents to manipulate in such a way that their performance can be guaranteed. We augment a probabilistic-bound from prior work with perception-only teams [8] to now capture important dynamics of perception-action teams vis-à-vis the maximum tracking error allowed for action actions to service targets denoted by perception agents. We leverage these upper-bound times into the high-level decision-making by explicitly modeling action durations, as per a Semi Markov Decision Process [223]. We design an attribute-based robot-interaction scheme between perception and action robots to increase the coordination resiliency and efficiency.

Our comprehensive, hierarchical framework includes a learning-based solution for MA-POSMDP, online EKF based target state estimation and uncertainty prediction at the higher level as well as a probabilistic upper-bound time setting for target service through teaming the perception and action robots at the lower level. Our approach enables interactive solutions for the high-level decision-making to explore and scan an unknown environment as well as the low-level control to coordinate co-dependent agents' actions with low-level performance guarantees. To our knowledge, this challenging set of problems and the development of such a coherent system have never been approached as a whole before; our paper is the first.

We empirically validate our framework in simulation and physical-robot implementation on the Robotarium multi-agent robot testbed [151]. Our evaluations demonstrate the efficacy and feasibility of our coordination framework by reducing the overall operation times in a benchmark wildfire-fighting case-study. We choose the application of wildfire fighting via heterogeneous, autonomous UAV as the case-study and motivating application for this work.

Contributions – The primary contributions of our work are:

1. Formulating a novel algorithm MA-SARTSA to learn a high-level decision making policy in an unknown, dynamic environment modeled by a MA-POSMDP.

2. Deriving an analytical upper-bound on tracking error that enables perception and ac-

tion agents to cooperatively determine whether action agents can provide a probabilistically-guaranteed service for a set of manipulation tasks given the state information received from perception agents for those tasks.

3. Proposing an attribute-based, individualized, coordination scheme between perception and action agents for an efficient, coordinated routing problem to set a state-of-the-art, outperforming our baselines in performance.

5.2.2 Problem statement and Formulation

Road Map

We first provide a high-level perspective on contributions and content provided in this work as a road-map listed as follows:

Step 1) We propose a novel formalization to describe the high-level decision making problem, namely MA-POSMDP and develop a novel variant of SARSA called MA-SARTSA as our learning-based solution (subsection 5.2.3). In this step, perception agents need to select among exploration (find new targets), exploitation (execute tasks with action agents by passing extracted target state information) or revisiting a previous target which has been acted upon via an action agent (re-examine targets). Re-examining a target is also dependent on the respective action agent's upper-bound service time ($\mathscr{T}_{\mathscr{U}\mathscr{B}}$) computed in Step 5 (see Figure 5.9).

Step 2) Considering the high-level exploitation and revisiting actions from Step 1, we begin the low-level planning and control by tackling the online target state estimation through EKF for perception agents (subsubsection 5.2.4). Here, we record measurement uncertainties to derive a tracking error upper-bound in Step 4.

Step 3) We provide our low-level scanning framework to individualize the interaction between perception and action robots to improve resiliency and cooperation efficiency (subsubsection 5.2.4). The objective of this step is to generate an action-agent-aware trajectory to account for the heterogeneity of robots and to make the probabilistic upper-bound service times calculated in Step 5 more accurate.

Step 4) Given EKF's covariance matrices from Step 2, we propose our uncertainty-based analytical tracking error upper-bound ($\text{TEB}^q_{\mathscr{T}_{\mathscr{U}\mathscr{B}}}$) which performs as a quantitative probabilistic performance guarantee for action agents which are incapable of sensing the dynamic targets. $\text{TEB}^q_{\mathscr{T}_{\mathscr{U}\mathscr{B}}}$ will then be used with the upper-bound service times for action agents, $\mathscr{T}_{\mathscr{U}\mathscr{B}}$, (subsubsection 5.2.4) in Step 5 to generate a set of target waypoints for these agents.

Step 5) We derive a set of probabilistic upper-bound service times $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ for action agents (subsubsection 5.2.4) to: (A) provide an upper-bound on the number of assigned tasks to a particular action agent (incapable of sensing) and guaranteeing that it will not miss the moving targets and (B) generate a time quantity as an input for perception agents' decision making in Step 1 (see Figure 5.9).

Problem Description and Formulation

To formulate the described dynamic optimization problem, consider a total of N_T dynamic targets with state-space $\left(s_t^{T_1}, \cdots, s_t^{T_{N_T}}\right) \in \{S_t^T\}^{N_T}$ (T represents target) and a known dynamics model, $\mathscr{M}_t$ (e.g., motion model introduced in Equation 3.1), but unknown parameter settings for the model (e.g., velocity as introduced in Equation 3.2). Further, consider a total of $N = N_P + N_M$ robots including N_P perception agents (i.e., sensing UAVs) [6] with state-space $\left(s_t^{P_1}, \cdots, s_t^{P_{N_P}}\right) \in \{S_t^P\}^{N_P}$ (P represents perception) and N_M action agents (i.e., manipulator UAVs) with state-space $\left(s_t^{M_1}, \cdots, s_t^{M_{N_M}}\right) \in \{S_t^M\}^{N_M}$ (M represents manipulation). Each robot state-vector is defined as $s_t^{P_i} \in \{S_t^P\}^{N_P}$ for perception (P) agents and $s_t^{M_i} \in \{S_t^M\}^{N_M}$ for manipulator (M) agents and contains robots' position, velocity and trait

[6] Terms "perception agent/robot" vs. "sensing UAV" and "action agent/robot" vs. "manipulator UAV" are used interchangeably throughout.

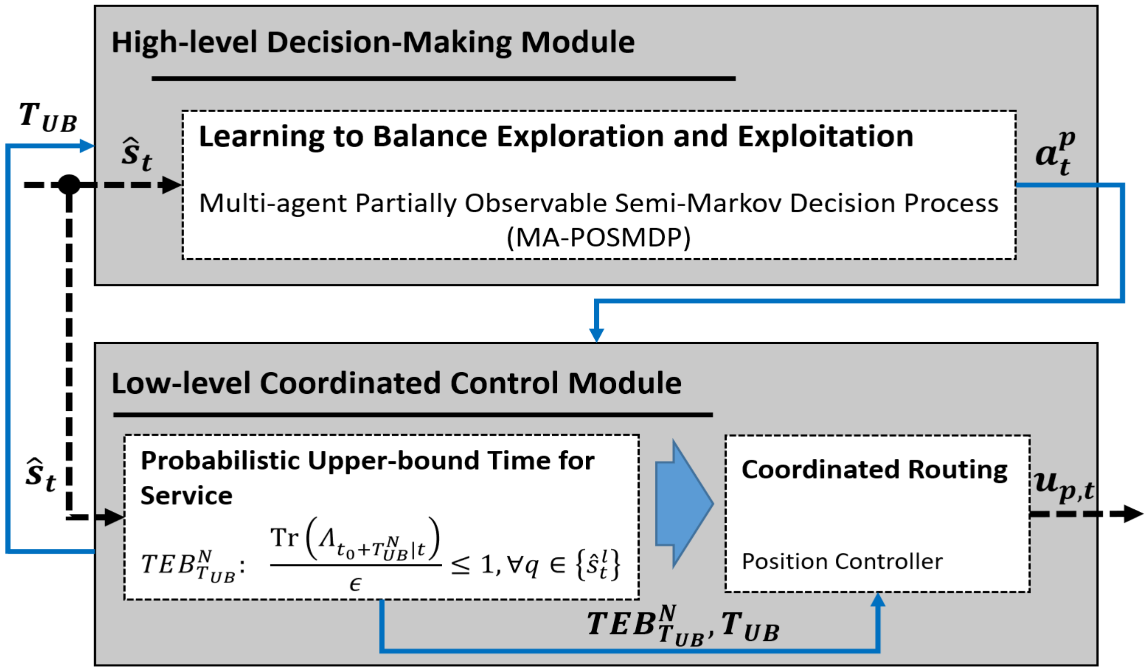

High-level Decision-Making Module
Learning to Balance Exploration and Exploitation
Multi-agent Partially Observable Semi-Markov Decision Process (MA-POSMDP)
T_{UB}
$\hat{s}_t$
a_t^p
Low-level Coordinated Control Module
Probabilistic Upper-bound Time for Service
$TEB_{T_{UB}}^N: \frac{\mathrm{Tr}\left(\Lambda_{t_0+T_{UB}^N|t}\right)}{\epsilon} \leq 1, \forall q \in \{\hat{s}_t^l\}$
Coordinated Routing
Position Controller
$u_{p,t}$
$TEB_{T_{UB}}^N, T_{UB}$

Figure 5.9: This figure depicts the proposed hierarchical coordination framework. Dashed links pass control input u_t to and receive state information $\hat{q}_t$ from robots.

information. Similarly, each target state-vector, $s_t^{T_i} \in \{S_t^T\}^{N_T}$, contains each target's position and velocity information alongside any other related, application-dependent state variable, such as R_t, U_t and θ_t in the fire propagation model.

We note due to the dynamicity of the environment (e.g., constantly state-changing targets), the problem resembles *restless decision making* problem [222]. In our restless decision-making problem, assuming $N_P < N_T$, at each time t, there will always be a state-changing target that is not covered by any perception agent. Here, regardless of the collective decisions of the perception-action team, the states of all observed or unobserved targets are constantly evolving through time. Conventionally, this restless problem would normally result in receiving two local *active* and *passive* rewards (i.e., r_t^{active} and $r_t^{passive}$ respectively), by the perception agent for both observed and unobserved targets [224]. However, here we combine these two rewards into a single overarching reward to encourage cooperation among perception agents (see subsubsection 5.2.3) through team-based rewards.

This problem can be considered as a variant of a Partially Observable Markov Decision

Process (POMDP). Accordingly, our objective is to find an optimal policy, π^*, over all admissible policies in set $\pi \in \Pi$, that maximizes the total expected, time-discounted reward accumulated by all perception agents over an infinite horizon, as in Equation 5.23. We provide detailed problem formulation and solution to this problem in subsection 5.2.3.

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \left(r_t^{P_1} + \cdots + r_t^{P_{N_P}} \right) \right] \tag{5.23}$$

For perception agent, P_i, if an action, $a_t^{P_i}$, at time t, is exploitation, it means the robot must extract the state-information from an already-found target point $s_t^{T_i} \in S_t^T$ to be passed to action agent M_j. Perception agents explore the environment and exploit the discovered targets to collectively generate a unified list of targets' estimated states, $\mathscr{L}_t = \{\hat{s}_t^1, \cdots, \hat{s}_t^{l(t)}\}$, in which $\hat{s}_t$ are targets' estimated state vectors. The length $l(t)$ changes with time, since exploring targets adds more exploitation options to the list and exploiting targets will remove targets from the list. We note that this variable-length state vector leads to a non-trivial dynamic-size state- and action-space representation problem in a learning-based decision making approach. We tackle this problem and propose a solution in subsubsection 5.2.3 - subsubsection 5.2.3.

To increase the efficiency of the composite robot team, we generalize the coordination problem such that action robots are capable of manipulating on more than one target during a single deployment. For example, an Airtanker UAV can fight the wildfire in more than just one grid spot. Accordingly, each vector in the list $\mathscr{L}_t$ is a set of estimated target states, $\hat{s}_t^l = \{\hat{q}_t^{m_1}, \cdots, \hat{q}_t^{m_c}\}$, in which m is the index of the manipulator agent and c is length of the waypoint list $\hat{s}_t^l$ sent to robot m. To determine the length, c, different factors such as targets' state transition model, $\mathscr{M}_t$ (Equation 3.1), action agent's motion/flight dynamics (e.g., maximum velocity, v_{max}, and turning bank, ω_{max}), and battery restrictions, need to be taken into account. We note that, due to dynamicity of the targets, simply passing the current coordinates of a sensed target from a perception agent to an action agent does not work. Accordingly, a probabilistic framework based on both target's motion model and action agent's dynamics is needed.

In the context of dynamic-target tracking, we need to make a high-level decision on whether to explore new targets or "exploit" known targets. Exploration implies searching the environment for new targets, while exploitation means examining an already found target to extract (estimate) necessary information for action agents and carry out the task by passing state-estimates to an action agent. When a perception robot, i, with position, $p_t^{P_i} \in s_t^{P_i}$, exploits an already-found target, initially, the closest action robot, j, with position, $p_t^{M_j} \in s_t^{M_j}$, is assigned to the task. To ensure the assigned action robot with current position, $p_t^{M_j}$, and maximum linear velocity, $v_{max}^{M_j}$, does not miss the moving target, τ, with estimated position, $\hat{q}_t^{\tau}$, and velocity, $\dot{\hat{q}}_t^{\tau}$, where $\left(\hat{q}_t^{\tau}, \dot{\hat{q}}_t^{\tau}\right) \in \hat{s}_t^{\tau} \in S_t^T$, we derive a measurement-uncertainty-based analytical probabilistic upper-bound time $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ (see Figure 5.9), which determines the upper-bound time required for the selected action agent to reach a close-enough proximity of the location of the sensed target. To this end, the new location, $\hat{q}_{t+\mathscr{T}_{\mathscr{U}\mathscr{B}}}^{\tau}$, that the moving target, $\hat{q}_t^{\tau}$, will move to, while the manipulator agent is on its way, needs to be estimated. To estimate $\hat{q}_{t+\mathscr{T}_{\mathscr{U}\mathscr{B}}}$, perception agents leverage EKF's multi-step prediction framework to propagate the detected point for $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ time into the future based on the target's motion model, $\mathscr{M}_t$, (e.g., fire propagation model in Equation 3.1).

Sensing robots continue to add targets to the set $\hat{s}_t^l = \{\hat{q}_{t+\mathscr{T}_{\mathscr{U}\mathscr{B}}}^{m_1}, \cdots, \hat{q}_{t+\mathscr{T}_{\mathscr{U}\mathscr{B}}}^{m_c}\} \in \mathscr{L}_t$, to be sent to a selected action robot. This process is performed through executing a feasibility test by comparing $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ to a maximum time allowable for each target track to propagate before the measurement uncertainty residual exceeds an acceptable predefined bound. The feasibility test, which we refer to as Tracking Error Bound (TEB) check, is preformed while specifically accounting for motion and battery restrictions of the assigned action manipulator robot, to jointly determine the length, c, of waypoint set, $\hat{s}_t^l$, and to generate executable trajectories within regions *reachable* by the action agent. See subsubsection 5.2.4 and subsubsection 5.2.4 for details.

If the TEB check is satisfied for a target point $\hat{q}_{t+\mathscr{T}_{\mathscr{U}\mathscr{B}}}$, the target is added to the waypoint set, $\hat{s}^{l}_{t}$, by perception agent and the robot then moves to execute the next action decision, a^{Pi}_{t}, made by the high-level decision-maker. The TEB check is performed continuously at each step for all the points on the waypoint list. This process continues until the first time the TEB check fails for a point or the overall travel time to visit all the nodes on the waypoint set reaches the assigned manipulator agent's battery limit. Now, the generated set of waypoints, $\hat{s}^{l}_{t}$, will be sent to the action agent. If the TEB check fails for the first point in a set, the perception agent rejects the assigned action agent and recruits another closer and/or faster robot to execute the task, if available (see subsubsection 5.2.4 and subsubsection 5.2.4 for details). For readers' convenience, we provide Table 5.2 listing the key variables used throughout the article. The following sections detail the steps in our framework.

5.2.3 High-level Decision-Making

As described in subsection 5.2.2, our objective in the high-level decision-making module is to automatically balance perception agents' effort in exploring the environment and exploiting found targets. To this end, we propose a learning framework termed MA-SARTSA learning to enable high-level decision-making in a MA-POSMDP. We note that while high-level decision-making directly governs only the perception agents' actions, manipulator agents indirectly contribute to this decision-making process because (1) action agents' dynamics and motion characteristics are explicitly considered by perception agents when generating the performance-guaranteed task trajectories (e.g., waypoint set of targets) for those action agents and (2) action agents execute their tasks (i.e., manipulate the target) in an exploitation action which can take an upper-bound service time of $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ to be executed[7].

[7]When high-level decision-maker chooses *exploitation* as perception agent's next action, action agents receive the estimated state-information from perception agents and then start executing their manipulation task, which takes an upper-bound service time of $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ to be executed.

Table 5.2: Summary of key nomenclature used in our paper.

Notation	Domain	Definition and Properties
N_T	$\mathbb{N}^+$	Total number of targets
N_P	$\mathbb{N}^+$	Total number of perception agents
N_M	$\mathbb{N}^+$	Total number of manipulator agents
$\hat{}$	$-$	Accent used for estimated variables
q^τ	$\mathbb{R}^{1\times 2}$	Location of τ-th firespot
p^{P_i}	$\mathbb{R}^{1\times 3}$	Position of i-th perception agent
p^{M_j}	$\mathbb{R}^{1\times 3}$	Position of j-th action agent
$\mathscr{L}_t$	$\mathbb{R}^{1\times l(t)}$	List of estimated waypoints at time t
$l(t)$	$\mathbb{N}^+$	Length of $\mathscr{L}_t$ at time t
$\hat{s}^l$	$\mathbb{R}^{1\times c}$	Estimated target state list; l-th element of $\mathscr{L}_t$
c	$\mathbb{N}^+$	Length of $\hat{s}^l$ at time t
$\mathscr{S}$	$\mathbb{R}^{w_1\times w_2\times k}$	State space; $w_1 \times w_2$ is the world size
$\mathscr{O}$	$\mathbb{N}^{w_1\times w_2\times k}$	Observation space; k is number of features
a	$\{0,1\}^{w_1\times w_2}$	Perception agent action representation
R	$\mathbb{R}$	Total accumulated discounted reward
$\mathscr{T}_{\mathscr{U}\mathscr{B}}$	$\mathbb{R}$	Upper-bound service time for an action agent
TEB^q	$\mathbb{R}$	Tracking-Error Bound (TEB) for target q
$\mathscr{E}$	$\mathbb{R}$	An acceptable threshold for TEB

Ground-Truth Environment Model

We assume that only the targets' motion model is known and do not assume any other prior information about targets, i.e. the number of targets or the parameters of targets' motion model. Thus, we are learning in a partially observable environment in which the state transition and the state transition times (as in Semi-MDP) may depend on unobserved variables. We describe the underlying ground-truth environment model in this section (which is not available to our agents) and move to agent's perspective (partial observability of states and limited set of action space) in subsubsection 5.2.3 and subsubsection 5.2.3.

As introduced in subsubsection 5.2.2, the environment's state consists of targets' states

$(s_t^{T_1}, \cdots, s_t^{T_{N_T}})$, perception agents' states $(s_t^{P_1}, \cdots, s_t^{P_{N_P}})$, and action (manipulator) agents' states $(s_t^{M_1}, \cdots, s_t^{M_{N_M}})$. Thus, by combining all involving state variables, the full state of the environment can be shown as in Equation 5.24.

$$s_t = (s_t^{T_1}, \cdots, s_t^{T_{N_T}}, s_t^{P_1}, \cdots, s_t^{P_{N_P}}, s_t^{M_1}, \cdots, s_t^{M_{N_M}}) \in \mathscr{S} = \{S_t^T\}^{N_T} \times \{S_t^P\}^{N_P} \times \{S_t^M\}^{N_M} \quad (5.24)$$

Under a fully-observed state (i.e., full observability), the action space for each perception agent is straightforward: $a_t^{P_i} \in A_t^P = \{$exploit target 1, exploit target 2, $\cdots$, exploit target $N_T\}$. There is no need for exploration action as we already know the state information in the fully-observable case. In the example of wildfire, each exploitation action consists of three phases. First, a perception agent travels to an exploration target. Second, the perception agent extracts fire state information and generates the performance-guaranteed path for the action agent (subsubsection 5.2.4). Third, an action agent drops fire retardant following the received path. The target however, may not be fully fulfilled (e.g., the fire may not be completely extinguished) by a single manipulator's execution, and thus the state transition $T(s'|s,a)$ is stochastic. Therefore, perception agents may choose exploitation again to revisit the target. If the target has been fulfilled, the corresponding exploitation action will be removed. If the target has not been fulfilled, the perception agent generates a new trajectory, and calls in an action agent again. The state transition time $F(s,a,s')$ is dependent on the upper-bound service time, $\mathscr{T}_{\mathscr{U}\mathscr{B}}$, for action agents that we derive in subsubsection 5.2.4. Unlike most SMDP formulations that have a reward rate during the entire transition time, in our setting, only an instant reward, r_t, is given on time t, when a target's task has been extinguished (e.g., extinguishing the fire at a target location). We argue that our instant reward setting is more semantically meaningful since longer execution on a target does not necessarily mean larger rewards. Upon completing the task for a target, the environment gives a reward proportional to the features of the target (e.g., a firespot's heat intensity). Thus, the total amount of reward is fixed when the initial state is given (the sum of all target importance). However, what we try to maximize is the discounted cumulative reward as per Equation 5.23. The temporal discount factor γ will encourage faster completion of the tasks.

We further note that through introducing both 1) a multi-agent setting and 2) an SMDP over the typical MDP, we create a more complex and more broadly applicable problem to tackle. For example, consider the scenario in which Agent 1 is assigned to exploit Target 1 while Agent 2 is assigned to execute on Target 2. The execution time of each action, as mentioned in the previous paragraph, is a random variable dependent on $\mathscr{T}_{\mathscr{U}\mathscr{B}}$. Therefore, the environment's state transition must consider the time ordering of agents' task executions. For the example above, if Agent 2 finishes first, the environment will transit to the state where Target 2 is processed and query the next action for Agent 2. Otherwise, the environment will, in turn, transit to the state where Target 1 is processed at first. We elaborate in more detail in subsubsection 5.2.3. We emphasize that including just the multi-agent or SMDP facets would not create this challenge. Instead, it is through the combination of both (i.e., a multi-agent SMDP) that causes this complexity.

Discrete-Event, Continuous Time Environment Simulation

In the majority of prior decision-making work [225, 226, 119, 227, 228], time is discretized by constant intervals (fixed-increment time progression) à la an MDP. However, discrete time-steps are not suitable for modeling our environment. Since we are in a multi-agent asynchronous execution setting, it is possible that before the current agent's action finishes, another agent finishes its task and queries for a new task, as described in the example in subsubsection 5.2.3. Therefore, we consider a discrete-event continuous-time progression for our environment that is more precise and efficient (see Figure 5.11).

In our setting, we define an event is when an agent finishes its current task and becomes idle. At such event, the decision-making algorithm needs to assign a new task to the agent given the current state. As the environment has perfect information about the targets, the robots' motion restrictions (e.g., maximum velocity and turning bank), and the upper-bound service time ($\mathscr{T}_{\mathscr{U}\mathscr{B}}$) (as derived in subsubsection 5.2.4), it could calculate the ground-truth time distribution for an assigned task. Accordingly, the simulation can readily determine the sequence of events, and thus, we can implement continuous time progress via discrete-event callback for the decision-making algorithm. In the discrete-event simulation, we can rewrite Equation 5.23 as $R = \sum_{t_i \in \mathscr{T}} \gamma^{t_i} r_{t_i}$, in which $\mathscr{T}$ represents a set of times for all discrete-events. The event set, $\mathscr{T}$, is important in that it bridges the continuous time process $\{s_t, a_t, r_t | t \geq 0\}$ and discrete chain $\{s_{t_i}, a_{t_i}, r_{t_i} | i \in \mathbb{N}\}$.

Observation and Action Space Representations

The agents and the underlying high-level decision-making algorithm do not have perfect information regarding the state and instead receive observations, $o \in \Omega$, which contain information about known targets. The action space available to each agent is not the entire target list since we have no prior information about the number of targets or target locations. Thus, we need to have an exploration action that is to find new targets. Further, the set of potential targets we could exploit is dynamic as exploration will result in uncovering new targets and thus increasing the target list length. Even with the simplifying assumption such that, for a single agent, exploration adds at most one new target, a conventionally defined action-space (e.g., discrete action set of choices to explore or which target to exploit) will be variable-length, and a canonical neural network cannot deal with such a dynamic length action space. Moreover, the agents also need to revisit previously exploited targets, as described in subsubsection 5.2.3.

Accordingly, we introduce a "landscape-based" action representation, $a \in \{0,1\}^{w_1 \times w_2}$ (illustrated in Figure 5.10), a $w_1 \times w_2$ binary matrix in which w_1 and w_2 represent the environment x and y dimensions. For exploitation action, matrix $\mathscr{A}$ is a one-hot encoding, and the corresponding 1 location on the terrain is the target location for exploitation. We further define an all 0 matrix representing exploration action. In this work, perception agents follow a predefined horizontal sweeping exploration strategy, but we also note the possibility

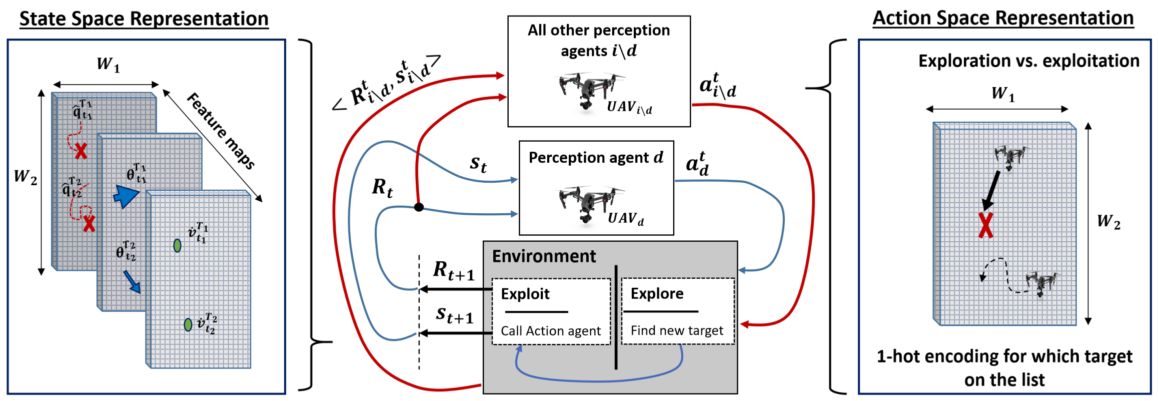

State Space Representation
w_1
w_2
Feature maps
All other perception agents $i\backslash d$
$UAV_{i\backslash d}$
$< R^t_{i\backslash d}, s^t_{i\backslash d} >$
$a^t_{i\backslash d}$
Perception agent d
UAV_d
s_t
R_t
a^t_d
Environment
R_{t+1}
s_{t+1}
Exploit
Call Action agent
Explore
Find new target
Action Space Representation
Exploration vs. exploitation
1-hot encoding for which target on the list

Figure 5.10: This figure depicts the designed, landscape-based, action and state space representation in the multi-agent Semi-MDP (middle). Our action-space (right side) can be represented as $\mathscr{A} \in \{0,1\}^{w_1 \times w_2}$, where an entry $\mathscr{A}_{i,j} = 1$ indicates the location of an *exploitation* action's target, and when $\mathscr{A}_{i,j} = 0, \forall i,j \in \mathbb{N}$ an *exploration* action is declared by the decision-making module. We take a similar approach to represent the state-space (left side) as overlaid feature maps forming a feature tensor $\mathscr{S} \in \{S_t\}^{w_1 \times w_2 \times k}$.

of learning an exploration policy under our proposed framework. With such representation, the action space $\mathscr{A}$ now has constant size $(1 + w_1 \times w_2)$, and semantic ambiguity will be avoided. We can dynamically generate a small subset of available actions each time with an observation. As shown in Figure 5.10, we take a similar approach to represent the observation-space as overlaid feature-maps forming a feature tensor of form $o \in \mathbb{N}^{w_1 \times w_2 \times k}$ where k is the number of features. The Observation space includes all targets that have been discovered so far, whether or not they are currently inside the respective UAV's FOV.

Multi-Agent Partially Observable Semi Markov Decision Process (MA-POSMDP)

We define a new problem setup termed MA-POSMDP as a 9-tuple $(\mathscr{S}, \Omega, O, \mathscr{A}, r, T, F, \gamma, \rho_0)$. State space $\mathscr{S}$ is introduced in subsubsection 5.2.3, and state $s_t \in \mathscr{S}$ consists of all the current information of the world for the process $\{s_t | t \geq 0\}$. Observation space, Ω, is introduced in subsubsection 5.2.3, and the current observation is given by the current state via observation model O: $o_t \sim O(\cdot|s_t)$. $\mathscr{A}$ is the action-space, and we leverage the landscape-based action representation described in subsubsection 5.2.3. We reiterate that despite the huge size of the entire action space, the possible action space under each state is very limited, being $l+1$ with l known targets and one explore action. $\gamma \in [0,1)$ is the temporal discount factor for each unit of time and $\rho_0(s)$ is the initial state distribution. $r(s,a)$ represents the reward when execution of action a on state s is finished, (i.e., an impulse function which has non-zero values only when an exploitation action is finished), and the amount of reward is relative to the priority/severity of the exploited target (e.g., fire intensity as in section 3.2 in the aerial wildfire fighting example). As such, we do not give explicit reward for exploration and discovering new targets, as our objective is defined on the team's discounted cumulative rewards instead of on each agent's individual rewards. This overarching reward is designed to encourage prolific cooperation between perception and action agents, and the learning mechanism, which will be introduced in the next section, will further enhance the collaboration. $F(s,a,s')$ is the time required for executing action a in state s and transit to s', which is determined by the introduced upper-bound time for service $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ and robots' dynamics (e.g., maximum velocity) in our proposed low-level component (subsubsection 5.2.4). Transition $T(s'|s,a)$ encodes the effect of agent's action a on state s and is also dependent on the time ordering as shown in subsubsection 5.2.3 and subsubsection 5.2.3. Note: we make the assumptions that agents do not fail and that the underlying motion model of the targets is known, and thus, the predicted duration time for actions are reliable.

Agents start with an initial belief over states, which, depending on the underlying application, could either encode a prior belief of some targets or be a non-informative prior. When a perception agent P_i is idle at time t, its current observation information $o_t \sim O(\cdot|s_t)$ is fed into the decision algorithm to query for an action a_t. The action is executed in the environment which calculates the next state $s' \sim T(\cdot|s_t,a_t)$, the required time for the action $f_t \sim F(s_t,a_t,s')$, and a reward $r_t \sim r(s_t,a_t)$. Since we are in a multi-agent, asynchronous execution setting, it is possible that before the current agent's action finishes, another agent finishes its task and queries for a new task. Therefore, we utilize a discrete-event simulation as described in subsubsection 5.2.3.

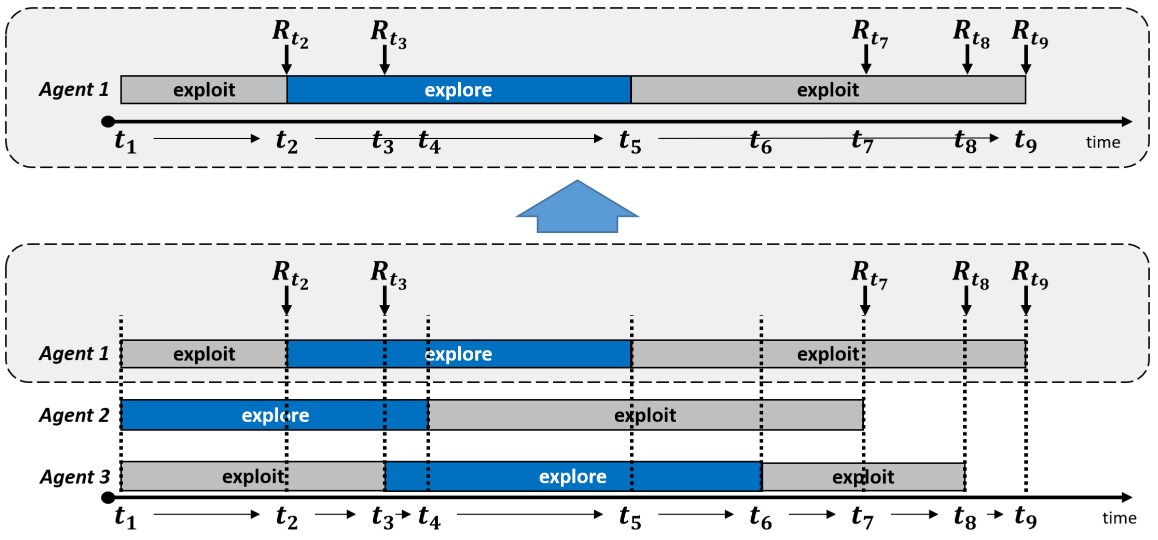


Figure 5.11: This figure depicts an example of timings and transitions in our discrete-event simulation of continuous time progression. The top gray panel represents the transitions in agent 1's perspective.

We rewrite our optimization goal in subsubsection 5.2.3 to introduce the optimization variable (policy π) in Equation 5.25:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathscr{R}(\pi) = \arg\max_{\pi \in \Pi} \mathbb{E}_\pi \left[\sum_{t_i \in \mathscr{T}} \gamma^{t_i} r_{t_i} \right] \tag{5.25}$$

Despite the similarity of the objective to standard MDP, we note that t_i in our MA-POSMDP setting in Equation 5.25 is a continuous value from a finite set $\mathscr{T}$ containing all events' times. We note that, our high-level decision making is a centralized process among perception agents. Our proposed MA-POSMDP learning process and the overall interaction between high-level and low-level modules are summarized in the Algorithm algorithm 4.

Multi-Agent SARTSA Learning

We first assume that all perception robots are homogeneous and can thus share the same decision-making module, specifically in this work, a NN. We note that heterogeneity still exists between perception agents and action agents. We extend the SARSA learning algorithm [126] for our particular MA-POSMDP formulation.

Algorithm 4: MA-POSMDP

```
 1: Initialize Global_Step= 0, Q-network $Q_\theta$ and target Q-network $Q^*_\theta$
 2: while not converged do
 3:     Obtain the upper-bound service time $\mathscr{T}_{\mathscr{U}\mathscr{B}}$, from one of the Equation 5.43,
        Equation 5.44 or Equation 5.47 for current task and manipulator agent
 4:     Obtain rollout $\tau = \{\langle t_i, s_{t_i}, a_{t_i}, r_{t_i}\rangle\}$ in which time is determined via $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ in line (3)
 5:     Transform rollout $\tau$ to single-agent perspective transitions $\langle s_{t_i}, a_{t_i}, \Delta t_i, r_{t_j}, s_{t_j}, a_{t_j}\rangle$ via
        Equation 5.29 and calculate $\mathscr{B}_i = \sum_{k=i+1}^{j} \gamma^{t_k - t_i} r_{t_k}$
 6:     Store $\langle s_{t_i}, a_{t_i}, \Delta t_i, \mathscr{B}_i, s_{t_j}, a_{t_j}\rangle$ to replay buffer
 7:     for i = 1 to iters do
 8:         Sample transitions $\langle s, a, \Delta t, \mathscr{B}, s', a'\rangle$ from replay buffer
 9:         Train $Q_\theta$ via Equation 5.30
10:         Global_Step ← Global_Step +1
11:         if mod(Global_Step, Update_Interval) == 0 then
12:             Update target network, $\theta^* \leftarrow \theta$
13:         end if
14:     end for
15: end while
16: return $Q_\theta$ =0
```

SARSA learning is based on the 1-step Temporal Difference (TD) signal in a typical MDP, relying on the transition tuple $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ as shown in Equation 5.26.

$$\delta_t^{\text{1-step TD}} = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{5.26}$$

SARSA's learning rule is given by Equation 5.27 in which α is the learning rate.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \delta_t^{\text{1-step TD}} \tag{5.27}$$

SARSA has also been extended to incorporate n-step TD error, which is a balance between 1-step TD estimation and Monte-Carlo (MC) estimation for the entire trajectory.

$$\begin{aligned}\delta_t^{\text{n-step TD}} &= r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n Q(s_{t+n}, a_{t+n}) - Q(s_t, a_t) \\ &= \left(\sum_{k=t+1}^{t+n-1} \gamma^{k-t-1} r_k\right) + \gamma^n Q(s_{t+n}, a_{t+n}) - Q(s_t, a_t)\end{aligned} \tag{5.28}$$

MC estimations update each state based on the entire sequence of observed rewards from the current state until the end of episode and 1-step TD learning performs the updates only based on the next-step reward. The N-step TD method, however, combines the two and applies updates based on observed rewards in next N steps.

We have two options to consider the SARSA-like transition tuples. First we can consider all the adjacent transitions, i.e. $\langle s_{t_i}, a_{t_i}, r_{t_{i+1}}, \Delta t_i, s_{t_{i+1}}, a_{t_{i+1}} \rangle$ in which $\Delta t_i = t_{i+1} - t_i$. However, the transition from s_{t_i} to $s_{t_{i+1}}$ does not necessarily come from a_{t_i}, as there might be other actions that finish before a_{t_i}. For example, in Figure 5.11, Agent 2's second action will transit to the end of Agent 1's second action; however, the consequent reward is not from Agent 2's action. Therefore, such a transition tuple will create an incorrect signal for learning algorithms which can be problematic.

Second, we can view transitions on a single agent's timeline as shown in Equation 5.29, where $j > i$ and, t_j and t_i are events for the same agent. Moreover, $j = \min_{k>i} k$, such that event k and event i are for the same agent and, $\Delta t_i = t_j - t_i$.

$$\langle s_{t_i}, a_{t_i}, \Delta t_i, r_{t_j}, s_{t_j}, a_{t_j} \rangle \tag{5.29}$$

For example, in the perspective of Agent 1, the transition of the second action is from t_2 to t_5. As such, the new state s_{t_j} (e.g. s_{t_5}) contains the influence of a_{t_i} (e.g., a_{t_2}). It could also contain other agents' action effects (e.g., a_{t_1} of Agent 2 and 3). However, we could view the process between t_i and t_j as executing the same policy π for $j-i$ times because all agents share the same policy. For instance, the transition between t_2 to t_5 could be viewed as applying the current policy three times, and we receive r_3, r_4, r_5 on t_3, t_4, t_5, respectively. Through such view, we obtain a novel TD signal (Equation 5.30) similar to the $j-i$ step TD error (Equation 5.28).

$$\delta_t^{\text{MA-TD}} = \left(\sum_{k=i+1}^{j} \gamma^{t_k - t_i} r_{t_k} \right) + \gamma^{\Delta t_i} Q(s_{t_j}, a_{t_j}) - Q(s_{t_i}, a_{t_i}) \tag{5.30}$$

For implementation, we propose to maintain a reward buffer $\mathscr{B}_i$ for each agent i to account for any reward the team has received collectively during the period when agent i is executing an action. For instance, $\mathscr{B}_1$ will record r_3, r_4 on t_3, t_4 during t_2 to t_5.

Thus, we could utilize $\delta_t^{\text{MA-TD}}$ to achieve SARSA-like learning by Equation 5.27 and all transitions in the view of each agent.

5.2.4 Low-level Coordinated Control and Planning

The lower-level in our hierarchical algorithm structure is a distributed, coordinated control and planning framework through which robots in the composite team will execute tasks (i.e., exploration or exploitation as assigned by the high-level decision-maker). The low-level control module is responsible for sequencing the tasks while accounting for robot's specific traits (capabilities) and motion restrictions. To this end, we first solve a coordinated routing problem between perception and action robots (subsubsection 5.2.4). Action robots are not capable of sensing the dynamic targets and, therefore, need to receive predicted future target states. Thus, to maximize target track quality, a probabilistic framework is developed in subsubsection 5.2.4 to propagate and evaluate measurement error residuals over time. We leverage the derived tracking error upper-bound (TEB) to compute the upper-bound time for action agents' service, $\mathscr{T}_{\mathscr{U}\mathscr{B}}$, in subsubsection 5.2.4 which is also used for the task timings in our MA-POSMDP formulation underlying the high-level decision-maker (subsubsection 5.2.3). $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ is the upper-bound time it takes for an action agent to finish its task (e.g., travel to fire locations and extinguish fire) after receiving the required state-information from the perception agent.

The rest of subsection 5.2.4 is organized as follows: in subsubsection 5.2.4, we tackle the problem of online target state estimation via UAV sensors (given the action from the high-level decision-maker decides exploitation) through EKF estimation. We investigate the time-dependency of EKF's measurement uncertainty in paragraph 5.2.4 as a prerequisite for validating our covariance-based planning. Then, in subsubsection 5.2.4, we propose our low-level scanning framework to improve resiliency and cooperation efficiency between perception and action agents. Eventually, we propose our analytical tracking error upper-bound in subsubsection 5.2.4 and derive a set of probabilistic upper-bound service times for action agents in subsubsection 5.2.4.

Preliminaries: Target State Estimation and Measurement Uncertainty Propagation via UAV Perception Agents

We utilize EKF as a suitable tool to both estimate the target states and to propagate the measurement uncertainty residuals due to the model and sensor inaccuracies [229, 230]. EKF locally linearizes the target's nonlinear motion model, $\mathscr{M}_t$ (e.g., the fire propagation model in Equation 3.1), and perception agent's observation model, $\mathscr{O}_t$ (**??**), about the estimate of the current mean and covariance, and thus $\mathscr{M}_t$ and $\mathscr{O}_t$ do not need to be linear functions of the state, but may rather be differentiable functions.

Considering a dynamic target on the ground, $q_t = [q_t^x, q_t^y]$, moving according to a nonlinear model, $\mathscr{M}_t$ (e.g., Equation 3.1), at each time, t, the observation mapping, $\mathscr{O}_t$, of a flying perception agent with pose, $p_t = [p_t^x, p_t^y]$ and altitude p_t^z, with respect to the target's location can be shown as in Equation 5.31.

$$\mathscr{O}_t : \{q_t, p_t\} \to \varphi_t : \|q_t - p_t\|_2 = \left\|q_t^z - p_t^z\right\|_2 \tan \varphi_t \tag{5.31}$$

Given $\mathscr{S}_{t-1}$ as the current joint state vector of the dynamic target and perception agent's states; desired is an estimated state vector, $\hat{\mathscr{S}}_t$, one step forward in time, given the current target position distribution, $q_{t|t-1}$, target's motion model with current parameters ($\mathscr{M}_{t|t-1}$), and perception agent's observation model of the target ($\mathscr{O}_{t|t-1}$). In other words, we seek to estimate the following joint Probability Density Function (PDF), ρ, in Equation 5.32.

$$\hat{\mathscr{S}}_t = \underset{\mathscr{S}_t}{\arg\max}\, \rho \left(q_{t|t-1}, p_{t|t-1}, \mathscr{M}_{t|t-1}, \mathscr{O}_{t|t-1}\right) \tag{5.32}$$

In the application of aerial wildfire monitoring where perception agents are tasked to seek out the state information of propagating firespots, we formulate this estimation with the EKF's state transition and observation equations as in Equation 5.33 and Equation 5.34.

$$\begin{bmatrix} \hat{\mathscr{S}}_t \end{bmatrix}_{8\times 1} = \begin{bmatrix} \left.\dfrac{\partial \mathscr{M}_t}{\partial \mathscr{S}_i}\right|_{\hat{\mathscr{S}}_{t|t-1}} \end{bmatrix}_{8\times 8} \begin{bmatrix} \mathscr{S}_{t-1} \end{bmatrix}_{8\times 1} + \omega_t \tag{5.33}$$

$$\begin{bmatrix} \hat{\Phi}_t \end{bmatrix}_{5\times 1} = \begin{bmatrix} \left.\dfrac{\partial \mathscr{O}_t}{\partial \Phi_i}\right|_{\hat{\Phi}_{t|t}} \end{bmatrix}_{5\times 8} \begin{bmatrix} \hat{\mathscr{S}}_t \end{bmatrix}_{8\times 1} + \nu_t \tag{5.34}$$

In these equations, $\mathscr{S}_t = \left[q_t^x, q_t^y, p_t^x, p_t^y, p_t^z, R_t, U_t, \theta_t\right]^T$ is the joint state vector, ω_t and ν_t are the process and observation noises, which are modeled by zero mean white Gaussian random variables, respectively, to account for stochasticity in fire behavior and inaccuracies in fire propagation and perception agents' observation models. $F_t = \partial \mathscr{M}_t / \partial \mathscr{S}_i$ is the state transition Jacobian matrix in which the FARSITE model introduced in section 3.2 is used as the transition model to derive the partial derivatives with respect to all of the state variables in $\mathscr{S}_t$. The observation Jacobian matrix, $H_t = \partial \mathscr{O}_t / \partial \Phi_i$, is a mapping model (see Figure 4.3) through which the predicted fire propagation model parameters and UAV locations are translated into a unified angle-parameter vector $\hat{\Phi}_t = \left[\varphi_t^x, \varphi_t^y, \hat{R}_t, \hat{U}_t, \hat{\theta}_t\right]^T$. We note that, although the angle-parameters, φ_t^y and φ_t^x, are complementary angles, we utilize both angles as our goal is to use these angles for tracking 2D pose uncertainty (and not just angle uncertainties). A detailed derivation of EKF equations, Jacobian matrices, and uncertainty propagation is provided in the Appendix A.

EKF Bayesian Posterior and Minimum Mean-Squared Error (MMSE) Estimate EKF is an approximate Bayesian filter, and taking the resulting Gaussian distribution as the true Bayesian posterior can be imperfect in some localization and tracking applications. Nevertheless, we note that, in this work, we do not require the true Bayesian posterior, and none of our planning/decision-making algorithms are based on a true Bayesian posterior.

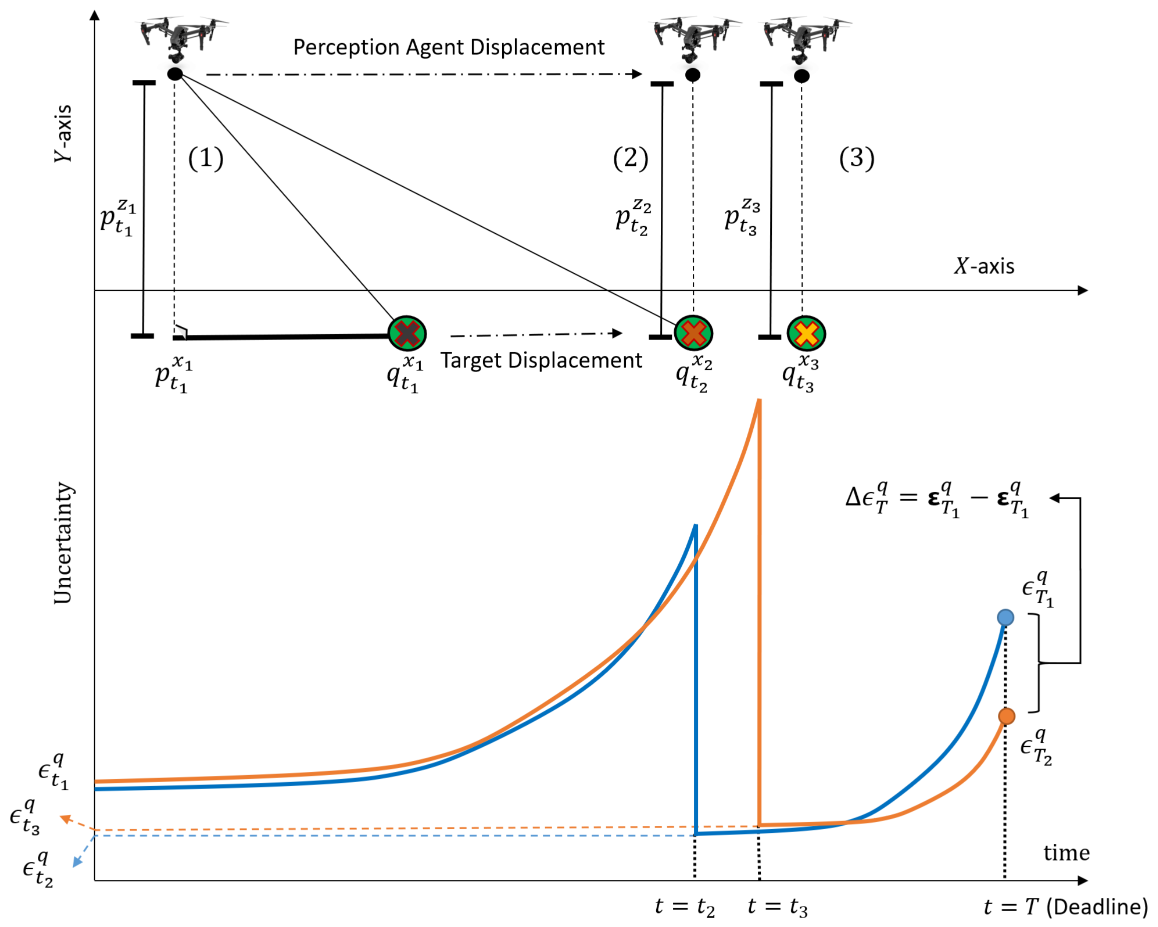


Figure 5.12: This figure depicts the time-dependency of measurement uncertainty as propagated by EKF. The dropping uncertainty regardless of the time of visit will always have the same value ($\varepsilon^q_{t_2} = \varepsilon^q_{t_3}$), if the observing robot's displacement to target is similar at both times.

Assuming a given map, $\mathscr{S}$ (e.g., a set of discrete world features or states), and a sequence of target relative observations, $\mathscr{Z}$, described by the conditional probability, $p\left(\mathscr{Z}|\mathscr{S},q\right)$, we seek to estimate the PDF of the variable q, as $p\left(q|\mathscr{S},\mathscr{Z}\right)$ through EKF. For this problem, it can be shown mathematically [231, 232] that if we parameterize the random vectors q and $\mathscr{S}$ with mean and variance, then the EKF will compute the MMSE estimate of the posterior. In our coordination framework, this MMSE estimate is an acceptable metric, in keeping with its use in prior localization and dynamic target tracking literature [203, 233, 234]. We leave improvements in state estimation and accurate target tracking to future work as they are not the focus of our current study.

Time-dependency of EKF's Measurement Uncertainty Our analytical upper-bound time for service in subsubsection 5.2.4 depends on the state-estimation measurement uncertainty; thus, we examine the time-dependency of the propagated error through the EKF. Considering the two examples presented in Figure 5.12 in which a perception agent starts at position (1) at time $t = t_1$ with some distance from a target and visits the moving target either at $t = t_2$ or $t = t_3$ (i.e. with $\Delta t = t_3 - t_2$ latency in latter case), we show that the dropping measurement uncertainty of the target's state, regardless of the time of visit, is only a function of the distance between the perception agent and the target. To this end, we first define the uncertainty drop in our scenario in Lemma 3 and in Theorem 4, we prove that such uncertainty drop is independent of time.

Lemma 3 *If the uncertainty (Kalman measurement residual) of a sensing robot observing a dynamic point q_t directly from distances $\Delta\mathscr{X}_{t_1}$ and $\Delta\mathscr{X}_{t_2}$ at times t_1 and t_2 (where $t_2 > t_1$) are defined by $\mathscr{E}^q_{t_1}$ and $\mathscr{E}^q_{t_2}$, respectively, then $\mathscr{E}^q_{t_2} < \mathscr{E}^q_{t_1}$ if and only if $\Delta\mathscr{X}_{t_2} < \Delta\mathscr{X}_{t_1}$. We define the uncertainty drop as follows in Equation 5.35.*

$$\Delta\varepsilon^q_{t_2,t_1} = \mathscr{E}^q_{t_2} - \mathscr{E}^q_{t_1} \tag{5.35}$$

Proof 4 A locally optimal strategy for state estimation is obtained by driving the robot to positions that maximize the prediction variance of the observation [147, 235, 236]. As such, minimizing the predicted state covariance (Equation A.1) corresponds to maximizing the covariance residual Λ_t in Equation A.2. According to Equation A.2, by setting the state covariance to identity ($\Sigma_{t|t-1} = I$) and keeping the noise covariance constant ($\Gamma_t = \Gamma$), we see that a maximally informative position for a robot is the one that minimizes $H_t H_t^T$. In other words, all other settings being equal, the closest possible position where dynamic observations change rapidly as a function of robot position [147] minimizes the determinant of the observation covariance and thus, minimizes the total uncertainty. Note that Λ_t is a function of both the state estimate and the map covariance $\Sigma_{t|t-1}$.

Next, we present Theorem 4 and a proof sketch. Please see Appendix A for a detailed proof of Theorem 4.

Theorem 4 *Measurement uncertainty drop about the states of a dynamic point q_t as defined in Lemma 3 and Equation 5.35, observed by a perception robot directly from a distance $\Delta\mathscr{X}$ at time T (i.e. $\mathscr{E}_T^q$), is independent of time. It is only a function of displacement $\Delta\mathscr{X}$ between the observer and the point.*

Proof 5 The model and observation measurement uncertainties associated with EKF estimation follow the general nonlinear uncertainty propagation law in Equation A.1 and Equation A.2 where $\Sigma_{t|t-1}$ is the predicted covariance estimate, $\Lambda_{t|t}$ is the innovation (or residual) covariance, F_t and H_t are the process and observation Jacobian matrices, and Q_t and Γ_t are the process and observation noise covariances, respectively.

$$\Sigma_{t|t-1} = F_t \Sigma_{t-1|t-1} F_t^T + Q_t \tag{5.36}$$

$$\Lambda_{t|t} = H_t \Sigma_{t|t-1} H_t^T + \Gamma_t \tag{5.37}$$

Considering Equation A.1-Equation A.2, changes in the uncertainty values occur through changes in the gradients in the process and observation Jacobian matrices (F_t and H_t). The gradients in the Jacobian matrices are calculated as derivatives of the target's motion model, $\mathscr{M}_t$ (Equation 3.1), and the perception agent's observation model, $\mathscr{O}_t$ (Equation 5.31), as in Equation 5.38, where $\mathscr{S}$ and Φ are the process and observation state-vectors, respectively.

$$F_t = \left[\left. \frac{\partial \mathscr{M}_t}{\partial \mathscr{S}_i} \right|_{\hat{\mathscr{S}}_{t|t-1}} \right] \quad \text{and} \quad H_t = \left[\left. \frac{\partial \mathscr{O}_t}{\partial \Phi_i} \right|_{\hat{\Phi}_{t|t}} \right] \tag{5.38}$$

Accordingly, matrix F_t is time-invariant if the gradients in this matrix are time-invariant, which depends on the target's motion model. Considering position q and velocity $\dot{q}$ of a moving target as the state variables, it can readily be seen that the gradients of a numerical motion model such as $q_t = q_{t-1} = \dot{q}_{t-1}\delta t$ (similar to Equation 3.1) with respect to its state variables are $\partial q_t / \partial q_{t-1} = 1$ and $\partial q_t / \partial \dot{q}_{t-1} = \delta t$ and are time-invariant for all constant time steps δt. Moreover, considering the observation model presented in Equation 5.31 and positions and velocities of the sensing UAV and the target as the state variables in Φ, the gradients in H_t are only functions of the Euclidean distance between the perception agent and the target locations. Accordingly, both F_t and H_t are time-invariant and with constant process and observation noise covariances (Q_t and Γ_t), the total uncertainty drop is also not a function of time. See Appendix A for a rigorous proof.

Low-level Control Framework for Scanning

The first step in our coordinated routing framework for the perception-action composite robot team is for the perception agents to search the environment and add newly found targets to a list. This list will be sent to an assigned action agent as its task. Such process of assigning tasks to action agents is generally known as a Multi-Robot Task Allocation (MRTA) problem [211]. We approximately solve this MRTA problem by tackling a constraint satisfaction problem with action agents as variables, tasks as domains, and constraints such as relative distance to the task locations, battery availability, etc. In our greedy solution, action agents are assigned to tasks based on their availability and their relative distance to the task location, since distance based assignments are fast and deployable in real-time.

Based on the underlying application, we leverage the CE-TSP with Steiner zone variable neighborhood search [204] so that the action agent only needs to get "close enough" to each goal, according to a predefined $\Delta-$disk proximity. In the application of aerial wildfire fighting, the perception agent observes multiple firespots within its Field-of-View (FOV); however, it does not need to pass all of these "nodes". Instead, the perception agent first identifies the overlapping $\Delta-$disks between k fire points as Steiner zones and then chooses the centroid nodes as coordinates to be added to the target list $\mathscr{L}_t$ [8].

While scanning the environment, perception agent takes into account the specific motion/flight model of the assigned manipulator robot. This process includes accounting for

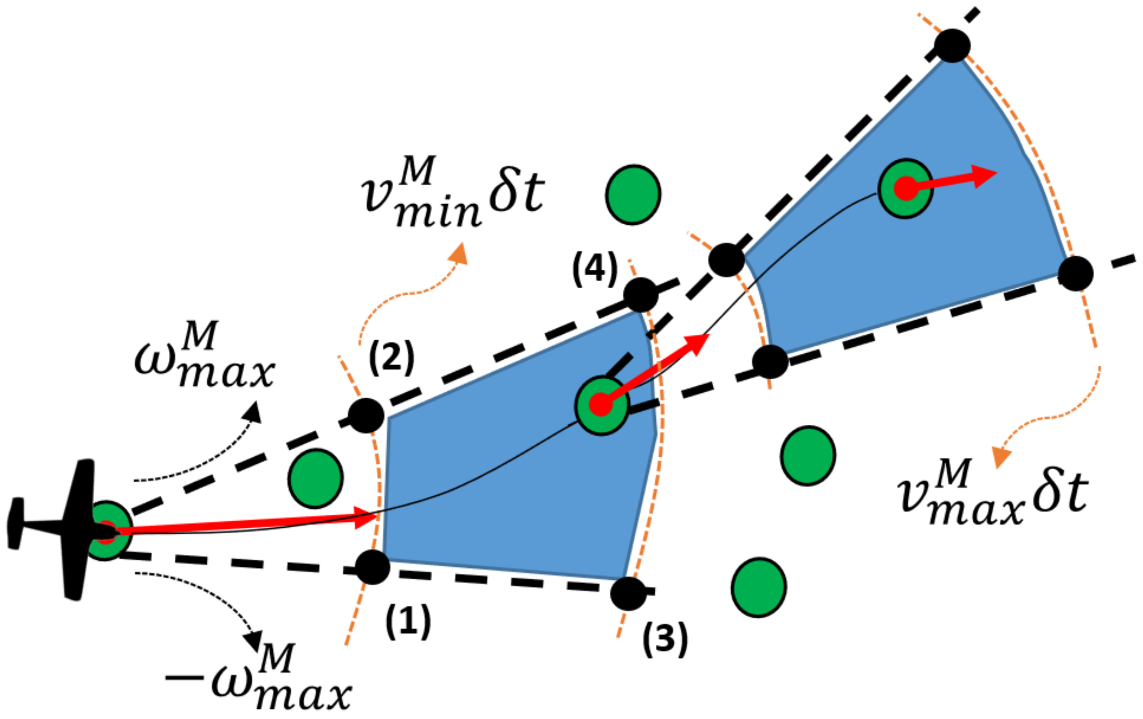


Figure 5.13: Perception agent accounts for the assigned manipulator agent's motion model, including its maximum turning rate ω_{max}^{M} (dashed black lines) and minimum and maximum linear velocities $\left[v_{min}^{M}, v_{max}^{M}\right]$ (red dashed lines). Sensing agent only scans the reachable areas for targets (green dots), enclosed by four vertices (1)-(4), (black dots).

the action agent's maximum turning rate ω_{max}^{M} and minimum and maximum linear velocities $\left[v_{min}^{M}, v_{max}^{M}\right]$. The perception agent leverages this information alongside the state-information of the first sensed target, which is propagated $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ steps into the future according to the target's motion model $q_{t+\mathscr{T}_{\mathscr{U}\mathscr{B}}}$, and explores for new targets only within reachable areas by the action agent. Note that $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ is the upper-bound time it takes for action agent M to travel to the target location. This process is elaborated in Figure 5.13 in which the blue areas represent the reachable polygons, the four vertex coordinates of which $\vec{p}_i$ s.t. $i = 1, \cdots, 4$ can be calculated through $\vec{p}_i = \vec{p}_M + \vec{u}_i$, in which $\vec{p}_M = q_{t+\mathscr{T}_{\mathscr{U}\mathscr{B}}}$ is the manipulator agent's position when it arrives at the propagated coordinates. Moreover, $\vec{u}_i$ represents the respective rotation vectors, rotating point $\vec{p}_M$ to $\vec{p}_i$ according to the manipulator agent's linear and angular velocities, which can be calculated according to Equation 5.39.

$$\vec{u}_i = \mathscr{D}\delta t \left(\frac{\vec{v}}{\|v\|}\right) R_z^{\varphi}\left((-1)^i \varphi\right) \tag{5.39}$$

In Equation 5.39, φ is the rotation-angle of the velocity-vector for one time-step, δt, due to maximum angular velocity and can be calculated as $\varphi = \omega_{max}\delta t/2$. $\mathscr{D}$ is the robot's planar displacement for one unit of time and equals $\mathscr{D} = v^M_{min}\delta t$ for $i = 1,2$ and is $\mathscr{D} = v^M_{max}\delta t$ for $i = 3,4$. Additionally, $R_z^{\varphi}(\varphi)$ is the rotation matrix around the z-axis. For a more detailed discussion of the above derivations, refer to Appendix B.

Leveraging the proposed scanning procedure results in an action agent-friendly path where a manipulator robot with motion restrictions (e.g., a fixed-wing aircraft) can directly visit all of the determined waypoints with no limitations.

Analytical Tracking-Error Bound

When exploiting a target, the perception agent estimates targets' states (e.g., position and velocity) to infer targets' motion dynamics and calculate two quantities: (1) the upper-bound time, $\mathscr{T}_{\mathscr{U}\mathscr{B}}$, required for an assigned manipulator robot to reach the sensed node, $\hat{q}$, and (2) a prediction of measurement residual covariance, $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ steps into the future by repeatedly applying EKF's prediction and update steps $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ times (Equation 5.33 - Equation 5.34). This process obtains an MMSE estimate of the Bayesian posterior (see paragraph 5.2.4). Moreover, $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ is the upper-bound time it takes for an action agent to finish its task after receiving the required state-information from perception agent.

Our tracking-error bound is inspired by the uncertainty residual ratio (URR) introduced in section 5.1 [7, 8]; however, our uncertainty-based temporal upper-bound derivations here are derived with a different goal in mind for a different underlying problem. In [7, 8] (section 5.1), we introduced an analytical uncertainty-based bound that compares the time it would take for a firespot to escape a perception agent's FOV relative to the total time it would take for that perception agent to complete a tour of observations on a set of distant nodes. In this work, however, we uniquely consider an analytical tracking-error bound that compares the time taken for a dynamic target's total uncertainty (e.g., error in target's state estimation) to grow to a predefined value, $\mathscr{E}$, versus the time it takes for the action agent to reach to the location of that target. In our application, this pre-defined value represents the maximum uncertainty over the target's state that would still allow for the action agent to effectively douse the fire target with retardant. To derive our bound, we leverage the model and observation measurement uncertainties associated with EKF estimation, shown below, where $\Sigma_{t|t-1}$ is the predicted covariance estimate, $\Lambda_{t|t}$ is the innovation covariance, F_t and H_t are the process and observation Jacobian matrices, and Q_t and Γ_t are the process and observation noise covariances.

$$\Sigma_{t|t-1} = F_t \Sigma_{t-1|t-1} F_t^T + Q_t \tag{5.40}$$

$$\Lambda_{t|t} = H_t \Sigma_{t|t-1} H_t^T + \Gamma_t \tag{5.41}$$

We note that this multi-step uncertainty propagation process only holds for Linear Time-Invariant (LTI) systems. In our framework, we utilize EKF, which locally linearizes the nonlinear motion model. Moreover, according to Theorem 4 (see paragraph 5.2.4), the measurement uncertainty as propagated by EKF is time-invariant. Now, we introduce an analytical TEB in Equation 5.42, in which t_0 is the current time, $\hat{s}_t^l \in \mathscr{L}_t$ is the l-th waypoint set of detected target coordinates shared among connected perception agents, q is the current node in $\hat{s}_t^l$, $\mathscr{E}$ is an acceptable error threshold and $\mathrm{Tr}(.)$ is the trace operation to sum the uncertainties of all state-variables in the residual covariance matrix.

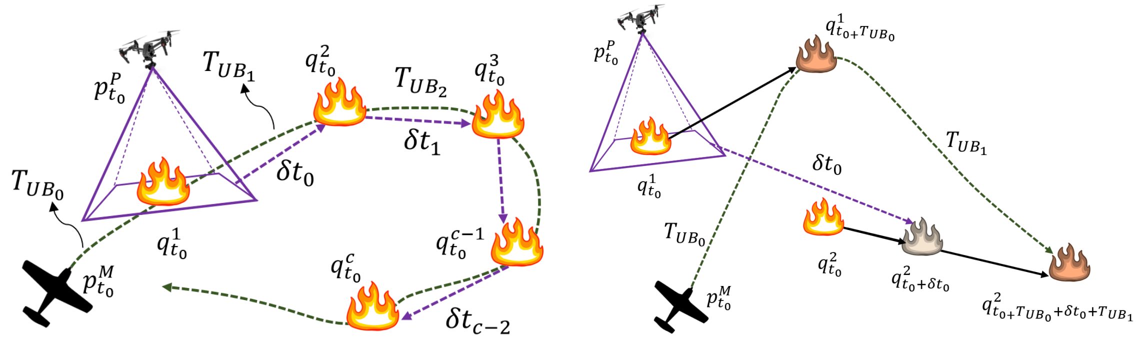
$$\mathrm{TEB}^q_{\mathscr{T}_{\mathscr{U}\mathscr{B}}} = \frac{\mathrm{Tr}\left(\Lambda_{t_0+\mathscr{T}^q_{\mathscr{U}\mathscr{B}}|t}\right)}{\mathscr{E}} \leq 1, \forall q \in \{\hat{s}_t^l\} \tag{5.42}$$

$\mathrm{TEB}^q_{\mathscr{T}_{\mathscr{U}\mathscr{B}}}$ is an indicator of the scale to which the action agent might miss the sensed dynamic target. A TEB greater than one demonstrates a quickly/stochastically moving target for which the chance of being missed by the action agent is high and vice versa. We note that selection of the parameter $\mathscr{E}$ is application-dependent, where it can be tuned to a small value for highly sensitive tasks and vice versa. The perception agent will use the TEB to determine the length, c, of the waypoint set, $\hat{s}_t^l = \{\hat{q}^{m_1}_{t+\mathscr{T}_{\mathscr{U}\mathscr{B}}}, \cdots, \hat{q}^{m_c}_{t+\mathscr{T}_{\mathscr{U}\mathscr{B}}}\}$, sent to the action agent m.

Probabilistic Upper-Bound Time for Service

In this section, we derive an analytical probabilistic upper-bound time, $\mathscr{T}_{\mathscr{U}\mathscr{B}}$, which determines the upper-bound time required for the assigned action agent to reach to a "close-enough" proximity to the determined coordinates and provide service (i.e., extinguishing a fire in aerial wildfire-fighting). $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ is directly modelling the state transition times, $F(s,a,s')$ (i.e., task duration), in our MA-POSMDP environment, as described in subsubsection 5.2.3. The number of waypoints in a set is chosen by determining the largest number of waypoints that can be included such that TEB (Equation 5.42) is satisfied at each timepoint for all targets captured by the waypoints. If the TEB bound does not hold for a target, the perception agent stops adding targets (e.g., target states) to the set and passes the generated trajectory to the action agent. The target's motion model is a key element in this upper-bound time. Accordingly, we derive three bounds, one for each of the following scenarios: (1) stationary targets, (2) dynamic targets, and (3) moving and spreading targets. These service upper-bound times are used both for calculating the TEB in Equation 5.42 in each scenario as well as in our high-level decision-maker as described in subsubsection 5.2.3. We follow a similar set of reasoning and assumptions as in section 5.1 and [7, 8].

Case 1: Stationary Targets The first scenario we consider involves multiple stationary targets. Figure 5.14a represents the coordination scheme for this case for two different types of UAVs as perception and action agents. Considering the time required for perception agent, P, to traverse between points n and $n+1$ represented as δt_{t_n}, we note that for a target with location q_{t_0} at time t_0, we have $q^n_{t_0} = q^n_{t_0+\delta_{n-2}}, \forall n \in \{\hat{s}_t^l\}^c$ (c is the length of l-th waypoint set)



(a) Case 1: Stationary Targets (b) Case 2: Dynamic Targets

Figure 5.14: This figure depicts collaborative planning for coordinating between perception and action agents (UAVs in this case). Figure 5.14a and Figure 5.14b represent the stationary and dynamic target scenarios, respectively. Note that the moving targets in the second case (black arrows) in Figure 5.14b, lead to a delayed exploitation by the perception UAV.

since targets do not move significantly in this case. Accordingly, the service upper-bound time for action agent, M, in Figure 5.14a to visit all c target locations in the l-th waypoint set can be calculated according to Equation 5.43. $p^M_{t_n}$ in Equation 5.43 is the n-th position of the action agent before travelling to the $(n+1)$-th target position. We note that, $p^M_{t_0}$, is known as action agent's current position and for the following targets we have $p^M_{t_n} = q^n_{t_n}$, that is, the action agent will move towards $(n+1)$-th target starting from the position of the n-th target, $q^n_{t_0}$. In Case 1, we have $q^n_{t_n} = q^n_{t_0}$.

$$\mathscr{T}^{(C1)}_{\mathscr{U}\mathscr{B}} = \sum_{n=0}^{c-1} \left(\frac{\left\| q^{n+1}_{t_0} - p^M_{t_n} \right\|}{\left\| v^M_{max} \right\|} \right) \tag{5.43}$$

Now, $\mathscr{T}^{(C1)}_{\mathscr{U}\mathscr{B}}$ will be checked by the perception agent in the TEB bound in Equation 5.42 to hold for each node at all times. A list of length c will be passed to the action agent if TEB is not satisfied for point $c+1$.

Case 2: Dynamic Targets The second scenario includes dynamic targets with a known motion model (e.g., Equation 3.1). In the case presented in Figure 5.14b, targets move, which leads to a delayed exploitation with δt_{t_n}. Consequently, the service upper-bound time for an action agent to get to the point c can be calculated according to Equation 5.44.

$$\mathscr{T}_{\mathscr{U}\mathscr{B}}^{(C2)} = \mathscr{T}_{\mathscr{U}\mathscr{B}_0} + \delta t_0 + \cdots + \mathscr{T}_{\mathscr{U}\mathscr{B}_{c-1}} + \delta t_{c-2} = \sum_{n=0}^{c-1} \left(\frac{\left\| q^{n+1}_{t_0 + \mathscr{T}_{\mathscr{U}\mathscr{B}_{n-1}}^{(C2)}} - p^M_{t_n} \right\|}{\left\| v^M_{max} \right\|} \right) \tag{5.44}$$

We note that, $p^M_{t_n}$ in Equation 5.44 has a similar reasoning as in Equation 5.43. In **??**, δt_{c-2} is the traverse time it takes perception agent, P, with position $p^P_{t_0}$ and maximum velocity v^P_{max} to get from point $q^{c-1}_{t_0}$ to point $q^c_{t_0}$ and can be computed as in Equation 5.45.

$$\delta t_{c-2} = \frac{\left\| q^c_{t_0 + \mathscr{T}_{\mathscr{U}\mathscr{B}_{n-1}}^{(C2)}} - p^P_{t_0 + \mathscr{T}_{\mathscr{U}\mathscr{B}_n}^{(C2)}} \right\|}{\left\| v^P_{max} \right\|}, \forall c \geq 2 \in \{\hat{s}^l_t\}^c \tag{5.45}$$

Note that $p^P_{t_0} = q^{c-1}_{t_0}$. Moreover, $q^{n+1}_{t_0 + \mathscr{T}_{\mathscr{U}\mathscr{B}_{n-1}}^{(C2)}}$ is the next target location to visit and can be obtained as shown in Equation 5.46.

$$q^{n+1}_{t_0 + \mathscr{T}_{\mathscr{U}\mathscr{B}_{n-1}}^{(C2)}} = \frac{q^{n+1}_t \left(v^M_{max}\right)^2 - p^M_{t_n} \left(\dot{q}^{n+1}_t\right)^2}{\left(v^M_{max}\right)^2 - \left(\dot{q}^{n+1}_t\right)^2} \tag{5.46}$$

Similar to Case 1, $\mathscr{T}_{\mathscr{U}\mathscr{B}}^{(C2)}$ is checked in TEB bound in Equation 5.42 to hold for each node at all times. The list will be cut at node c if TEB does not hold for point $c+1$.

Case 3: Moving-Spreading Targets The third scenario involves the case of targets that move and spread. This case is of particular interest in applications such as aerial wildfire fighting and oil-spill control in oceans via composite teams of robots. Here, single nodes of targets expand over time and thus can escape the perception agent's FOV. Assuming the spreading target's centroid as a dynamic node, the service upper-bound time, $\mathscr{T}_{\mathscr{U}\mathscr{B}}$, for the action agent in Case 3 can be derived through a procedure similar to Case 2 and thus $\mathscr{T}_{\mathscr{U}\mathscr{B}}^{(C3)} = \mathscr{T}_{\mathscr{U}\mathscr{B}}^{(C2)}$ holds. Nevertheless, unlike the other two cases, $\mathscr{T}_{\mathscr{U}\mathscr{B}}^{(C3)}$ cannot be used directly in the TEB bound in Equation 5.42 to generate the waypoint set. We note that as targets grow out of the perception agent's FOV, a new Steiner area in our CE-TSP framework will be created and thus, a new separate waypoint will form, leading to losing track for the centroid location of the current target. Accordingly, an additional condition monitoring the target's spread needs to be checked in addition to TEB to determine the length c of waypoint set in Case 3, as defined in Equation 5.47.

$$\mathscr{T}^{(C3)}_{\mathscr{U}\mathscr{B}_c} \leq \mathscr{T}^{c}_{max}, \forall c \in \{\hat{s}^l_t\}^c \tag{5.47}$$

Equation 5.47 is designed to check if the time it takes a spreading target, c, to escape the perception agent's FOV, ($\mathscr{T}^{c}_{max}$), is greater or smaller than the upper-bound service time, $\mathscr{T}^{(C3)}_{\mathscr{U}\mathscr{B}_c}$, of the action agent. As such, while the above condition holds, the process continues as before and a set of size c will be passed to action agent if TEB is not satisfied for point $c+1$. The escaping time $\mathscr{T}^{c}_{max}$ can be estimated as shown in Equation 5.48, following the outline of [8].

$$\mathscr{T}^{c}_{max} = \frac{-\beta + \sqrt{\beta^2 - 4\mu\delta}}{2\mu}, \forall c \in \{\hat{s}^l_t\}^c \tag{5.48}$$

The parameters μ, β and δ in Equation 5.48 can be calculated as $\mu = \frac{4|\hat{s}^l_t|_t \dot{q}^2_t}{\mathscr{W}_t v^P_{max}}$, $\beta = 1 + \frac{2|\hat{s}^l_t|_t \dot{q}_t}{v^P_{max}}$ and $\delta = \frac{2\Delta L_t}{v^P_{max}\left(1 - 2\dot{q}_t\left(|\hat{s}^l_t|_t - 1\right)\right)}$, in which $|\hat{s}^l_t|_t$ is the length of the waypoint set at time t, $\mathscr{W}_t$ is the current width of the perception agent's FOV, $\dot{q}_t$ is the target's current total velocity in XY-coordinates and $\Delta L_t = \sum_{n=0}^{c-1} \left\| q^{n+1}_t - q^n_t \right\|$ is the total travel distance for the action agent between the last node added to the l-th waypoint set $\hat{s}^l_t$ and the first node. A detailed derivation for Equation 5.48 can be found in paragraph 5.1.4.

Finally, to account for action UAVs' battery constraints, $T^{B_M}_t$, we directly compare the service upper-bound times, $\mathscr{T}_{\mathscr{U}\mathscr{B}}$, calculated in Equation 5.43 and Equation 5.47, with the remaining battery-life of action agent M at time t, by updating $\mathscr{T}_{\mathscr{U}\mathscr{B}}$ as $\mathscr{T}_{\mathscr{U}\mathscr{B}} \leftarrow$ minimum $\left(\mathscr{T}_{\mathscr{U}\mathscr{B}}, T^{B_M}_t\right)$. The output of this update will be used in Equation 5.42 to propagate the uncertainty of the target locations at most as long as it will take the action agent to

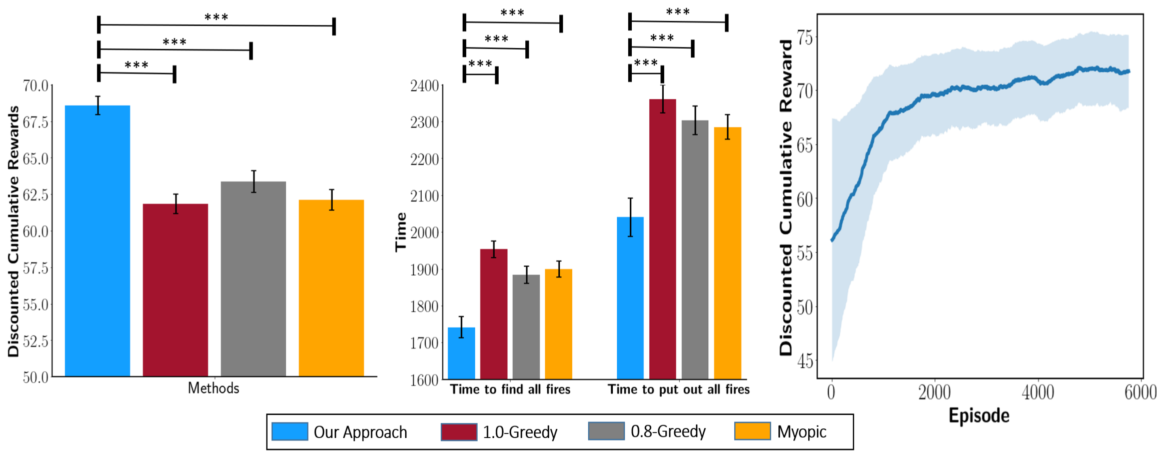


Figure 5.15: This figure depicts our benchmark comparison results. The leftmost figure shows a comparison on reward between our approach and benchmarks. Test repeated for 100 times. Error bars are standard error. The middle chart depicts a comparison on time (unit time in simulation) taken to find all fires and put out all fires. Test repeated for 100 times. Error bars are standard error. The rightmost figure shows our RL agent's training reward over the course of training averaged over three runs, where the shaded region depicts the standard deviation.

provide service or its battery-life allows it to operate.

5.2.5 Empirical Evaluation

In this section, we empirically evaluate both our high-level decision making and low-level coordinated control modules in an aerial wildfire fighting case-study, in which teams of heterogeneous, autonomous UAVs are tasked to work together to fight a propagating wildfire efficiently, with limited resources, such as the number of available UAVs and battery-lives. Moreover, we perform several experiments to evaluate the scalability and computational efficiency of our framework as well as its compatibility to different exploration strategies.

In this particular simulation, we generated ten different terrains of 100×100 with 30 initial firespots of different intensities. The firespots move and propagate according to the FARSITE model introduced in section 3.2. The parameters of FARSITE model R_t (spread rate), U_t (wind velocity) and θ_t (wind azimuth) are initialized as normally distributed random variables with $\langle \mu_R = 6, \sigma_R = 3 \rangle$, $\langle \mu_U = 5, \sigma_U = 2 \rangle$ and $\langle \mu_\theta = \pi/4, \sigma_\theta = 1 \rangle$, respectively.

Each UAV, d, is assigned a linear velocity range of $[v^d_{min}, v^d_{max}]$ and a maximum angular velocity, ω^d_{max}. Each class of UAVs (e.g., fixed-wing or quadcopter) has its own special characteristic, for instance for fixed-wing UAVs $v_{min} \neq 0$ while for quadcopters, $v_{min} = 0$ and $\omega_{max} = \infty$. All mobile robots in this environment are modeled as Dubins vehicles. In our experiments, we only assign quadcopters to perception agents and fixed-wing aircraft to action agents.

Each perception UAV is controlled by a shared NN trained via our MA-SARTSA learning algorithm (subsubsection 5.2.3). The policy network architecture consists of two fully connected layers (hidden units are 128 and 32, respectively) with Rectified Linear Unit (ReLU) activations. Fire intensity increases if a firespot is left unexploited. The NN represents the Q-function, $Q_\theta : \mathscr{O} \times \mathscr{A} \rightarrow \mathbb{R}$, inputting observation and action and outputting the corresponding Q-value. For discrete action spaces, we obtain the policy via $\pi = \arg\max_{a \in \mathscr{A}} Q_\theta(o,a)$. If a firespot is exploited (i.e., a perception agent travels to it, extracts its state information, generates a guaranteed task for action agent (only for the one target), and an action agent executes the task), fire intensity drops as a result of the action agent's firefighting service. Rewards, which are equal to the initial fire intensity, will only be given when the fire is extinguished. The learning rate, α, temporal discount factor, γ, and training epoch number were chosen empirically to be 0.001, 0.999 and 6000, respectively. Figure 5.15 (right) demonstrates the RL agent's cumulative reward throughout training and indicates success.

Benchmarks

We compared our approach with two categories of heuristic benchmarks: (1) ε-greedy and (2) myopic, as described below:

- **ε-greedy:** The perception agents on the composite team exploit a firespot with probability ε and choose a known firespot according to uniform distribution. With probability $1-\varepsilon$, agents choose to explore the terrain to discover new firespots. As an extreme case, when $\varepsilon = 1$, agents will only explore when there is no known fire to exploit.

- **Myopic:** The perception agents on the composite team always exploit a fire immediately after discovering it until the fire is extinguished by the action agent. When the current fire is extinguished, the agent will choose to explore. Myopic agents do not cooperate as each agent only exploits a fire that the agent has found.

Evaluation Results

To evaluate the performance against benchmarks, we considered three metrics: (1) the discounted cumulative reward (our learning algorithm's objective), (2) the time required for agents to find all the firespots, and (3) the time required for the team to extinguish all firespots (i.e., the perception-action cooperation objective). The results of the evaluation over 100 trials of simulation are presented in Figure 5.15. For all three evaluation metrics, our approach outperforms the baselines.

We performed statistical tests to show our approach's superior performance on all three metrics. We tested for normality and homoscedasticity and did not reject the null hypothesis for rewards and time to find all fires using Shapiro-Wilk ($p > .2$) and Levene's Test ($p > .2$), but we reject normal hypothesis for time to extinguish all fires. One-way ANOVA showed a significant difference between four groups on discounted cumulative rewards earned and time to find all fires ($p < .001$). Tukey's posthoc tests showed a significant difference between our approach and all benchmarks. Kruskal-Wallis test showed a significant difference between the four groups on time to extinguish all fires ($p < .001$), and posthoc tests showed also showed significant improvements.

Scalability and Sensitivity Analysis – In our scalability experiments, we tested the performance of our MA-SARTSA algorithm over various environmental and algorithmic parameters. We tested the scalability of our algorithm on a set of different number of agents in the composite team (5, 8 and 11 agents) and number of propagating firespots in the terrain (10, 30 and 50 firespots). We computed the accumulated mean reward ($\pm$ Standard Deviation

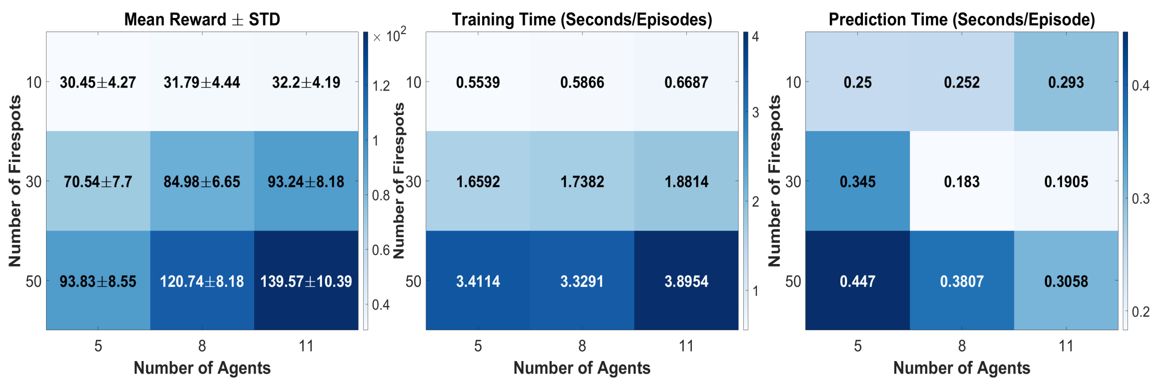


Figure 5.16: This figure depicts our scalability and computation time results. The leftmost figure shows the mean reward accumulated in nine different scenarios specified by the number of agents in the composite team and the number of firespots in the environment. Test repeated for 10 times and standard deviations are presented (± STD). The middle and rightmost charts depict the training and prediction times for the same nine scenarios, respectively. The times are shown as seconds per episodes.

(STD)) as well as the training and prediction times across nine different environment setups. The result are represented in Figure 5.16. As shown, the training time increases as the size of fire and number of agents increase, while the prediction time decreases when the number of agents increases. Moreover, the accumulated average reward also increases significantly when the number of agents and firespots increase. Accordingly, our framework scales to the number of firespots and expectedly can achieve higher performance when the number of agents increases. We also tested the sensitivity of our framework to different values of agents' FOV size as a key parameter, which has a direct correlation with the $\Delta-$disk proximity in our CE-TSP approach. Again, we evaluated the performance by computing the mean rewards (±STD) and computation times as described above, on our original environment setup described in subsection 5.2.5. As represented in Table 5.3, by increasing the size of agents' FOVs from a 1×1 grid (e.g., only an agent's location) to a 5×5 grid (e.g., two-hop neighborhoods around agents), our framework is capable of achieving similar results and thus, shows low sensitivity to values of FOV and $\Delta-$disk.

We also evaluated our framework under different exploration approaches: (1) horizontal sweeping, (2) diagonal sweeping and (3) spiral paths. The goal was to investigate the

Table 5.3: Results of our Sensitivity experiment for different values of agents' FOV sizes. Training and predictions times unit is $[\frac{Sec.}{Ep.}]$.

FOV	**Reward (μ±std)**	**Training Time**	**Prediction Time**
1×1	69.97±6.12	1.7816	0.187
3×3	70.38±5.37	1.7962	0.36
5×5	70.06±7.07	1.8300	0.326

Table 5.4: Results of our performance sensitivity evaluation under horizontal sweeping, diagonal sweeping and spiral exploration paths. Training and predictions times unit is $[\frac{Sec.}{Ep.}]$.

Method	**Reward (μ±std)**	**Training Time**	**Prediction Time**
Horizontal	69.97±6.12	1.7816	0.187
Diagonal	61.46±7.34	1.7882	0.537
Spiral	68.21±7.95	1.9432	0.445

sensitivity of the framework to the exploration paths taken by agents. As in our environment setup from subsection 5.2.5 (e.g., five agents and 30 initial firespots), we computed the total average rewards, as well as the training and prediction times for comparison between exploration methods, as shown in Table 5.4. Our framework achieves similar performances in terms of average total reward and computation times under different exploration methods, demonstrating low sensitivity to the exploration approach.

5.2.6 Demonstration: Multi-Robot Testbed

We evaluated our low-level coordination module on physical robots to directly include robot's motion dynamics and to test the calculated timings. The coordinated control and planning module was implemented and tested in the Robotarium multi-robot platform [151] on two simple (dynamic targets with linear motion models) and complex (propagating wildfire with numerous dynamic firespots) scenarios. Each case is elaborated below.

Demonstration Scenario (A) - Dynamic Targets with Linear Motion Models:

For this case, six dynamic targets (one target is intentionally left stationary, $\dot{q} = 0$) are randomly initialized in a terrain. Targets are assumed to have a linear motion model (moving on a straight line) with constant velocities, and their locations are initially unknown to four robots. The composite robot team includes two perception and two action robots that are required to stop the targets from moving. Accomplishing this task can be achieved by moving an action robot to the location of the respective target; however, since action agents are unable to see targets, a collaboration between perception and action robots is required. We apply our coordination framework to enable this collaboration. Excerpts of this experiment are depicted and described in Figure 5.17. As a result, our composite robot team successfully discovered (by perception robots) and stopped (by action robots) all six initially unknown targets collectively, without losing the track of any targets. The video recording of this experiment can be found in the first part of the provided supplementary video as well as at `https://youtu.be/qcomKuD-Hhw`.

Demonstration Scenario (B) - Propagating Wildfire with Numerous Dynamic Firespots:

In this case, we evaluated our perception-action robot team coordination module in a more complex aerial wildfire-fighting environment. Four randomly initialized distinct fire areas are considered to be explored by two perception robots. FARSITE wildfire propagation model has been leveraged, and the model parameters are initialized as detailed in subsection 5.2.5. We note that in our experiments we use omnidirectional robots; as such, we do not need to leverage our scanning framework (subsubsection 5.2.4) and perception agents pass propagated centroid nodes of discovered groups of firespots to three action robots. The video recording of this experiment can be found in the second part of the provided supplementary video as well as at `https://youtu.be/qcomKuD-Hhw`. For comparison, an embedded simulated video is added on the top-left of the screen to demonstrate the wildfire propagation simulation with similar parameters but without the composite team attempting

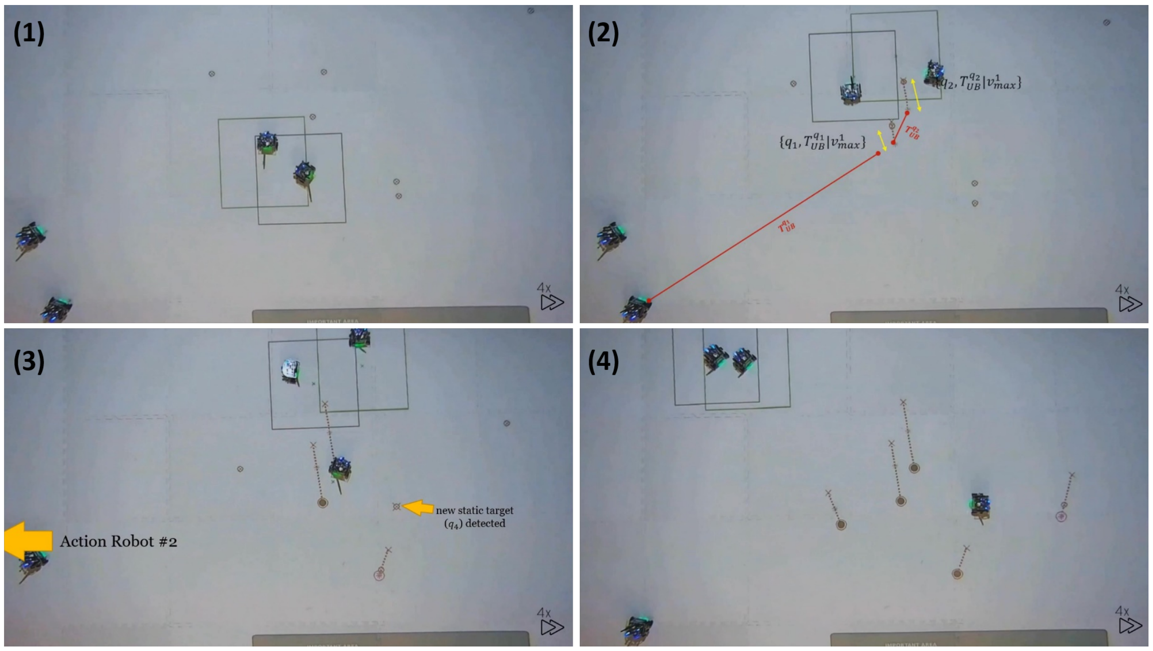
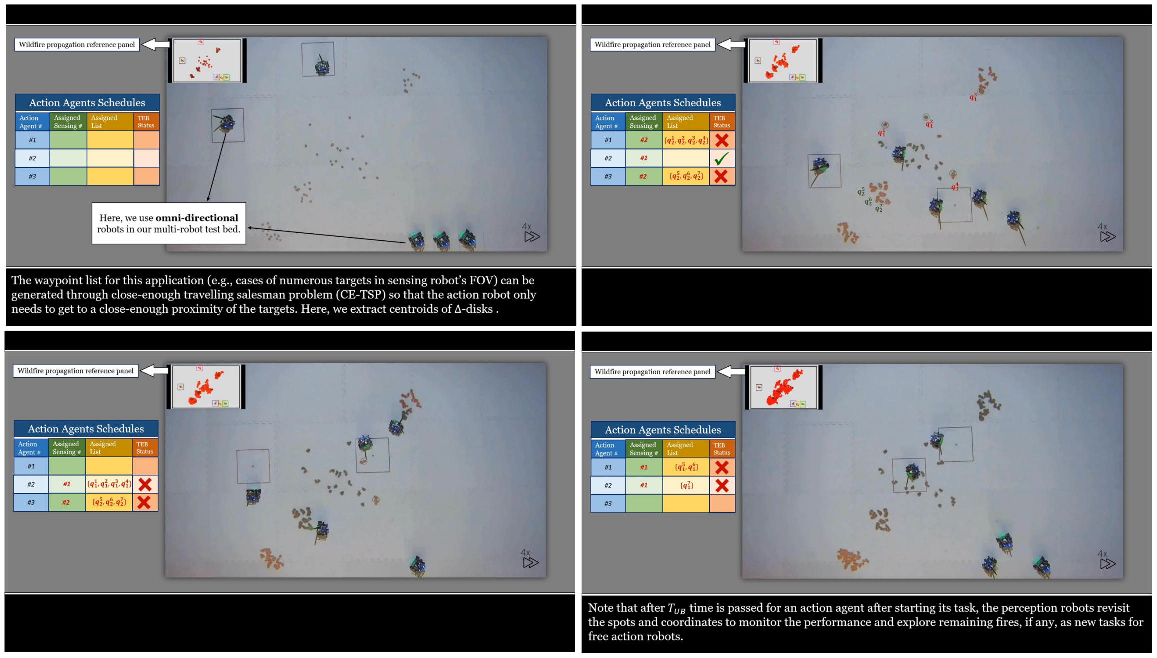

(1)
4x
(2)
$\{q_2, T_{UB}^{q_2}|v_{max}^1\}$
$\{q_1, T_{UB}^{q_1}|v_{max}^1\}$
$T_{UB}^{q_1}$
4x
(3)
new static target (q_4) detected
Action Robot #2
4x
(4)
4x

Figure 5.17: (1) Two perception robots start exploring the environment for targets. (2) Perception robots evaluate the TEB bound for an action robot. (3) The action robot starts the task the first time TEB is violated; perception robots continue to explore. (4) Final standing of the robot team after servicing all six initially unknown targets.

to fight the wildfire. As a result, our composite robot team successfully performed the complex task of autonomous wildfire fighting by effectively coordinating. Figure 5.18 illustrates four sample steps (initial standing to final standing) of the mentioned process.

5.2.7 Discussion

Our empirical evaluations demonstrate the feasibility of our framework and its superior performance in all three evaluation metrics stated in subsubsection 5.2.5, as compared to the employed benchmarks. For the epsilon-greedy agent, we tested various epsilons, $\varepsilon \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ where $\varepsilon = 0.8$ obtained the best result among all the ε-greedy agents; yet, our MA-SARTSA agent outperformed the 0.8-greedy agent, as well as the greedy agent (i.e., 1-greedy). We can see from the middle bar-plot in Figure 5.15 that the 0.8-greedy agent spends slightly less time to find all of the fire spots as compared to the Myopic agent, while on the other hand, the Myopic agent spends slightly less time to



Action Agents Schedules

Action Agent #	Assigned Sensing #	Assigned List	TEB Status
#1			
#2			
#3			

Here, we use **omni-directional** robots in our multi-robot test bed.

The waypoint list for this application (e.g., cases of numerous targets in sensing robot's FOV) can be generated through close-enough travelling salesman problem (CE-TSP) so that the action robot only needs to get to a close-enough proximity of the targets. Here, we extract centroids of Δ-disks .

Action Agents Schedules

Action Agent #	Assigned Sensing #	Assigned List	TEB Status
#1	#2	$\{q_2^1, q_2^2, q_2^3, q_2^4\}$	✗
#2	#1		✓
#3	#2	$\{q_2^5, q_2^6, q_2^7\}$	✗

Action Agents Schedules

Action Agent #	Assigned Sensing #	Assigned List	TEB Status
#1			
#2	#1	$\{q_1^1, q_1^2, q_1^3, q_1^4\}$	✗
#3	#2	$\{q_2^5, q_2^6, q_2^7\}$	✗

Action Agents Schedules

Action Agent #	Assigned Sensing #	Assigned List	TEB Status
#1	#1	$\{q_1^5, q_1^6\}$	✗
#2	#1	$\{q_1^7\}$	✗
#3			

Note that after T_{UB} time is passed for an action agent after starting its task, the perception robots revisit the spots and coordinates to monitor the performance and explore remaining fires, if any, as new tasks for free action robots.

Figure 5.18: This figure presents four sample demonstrations of our coordinated control and planning framework implemented on physical robots (initial to final standings) in the complex scenario (B) described in subsubsection 5.2.6 (`https://youtu.be/qcomKuD-Hhw`).

extinguish all of the fires than the 0.8-greedy agent. Accordingly, none of the Myopic or ε-greedy policies can be concluded to be an empirically better policy than the other. The MA-SARTSA learning algorithm successfully helps the policy to maximize the objective in Equation 5.25 as is shown in the learning curve in Figure 5.15 (right). The figure shows that within 2000 episodes, the algorithm has already been able to find a much better policy than its initial policy, illustrating the empirical success of MA-SARTSA and sample efficiency.

Nevertheless, the significant improvement of our MA-SARTSA framework over both the Myopic and the ε-greedy agents shows that the perception agents must make decisions based on not only the state information acquired from the environment, but also according to their collaborator action-agents' characteristics and motion dynamics. Myopic and ε-greedy agents do not model the time it takes for action agents to finish their tasks. Our MA-SARTSA, however, incorporates action agents' upper-bound service times (e.g., the time it takes for action agents to finish their tasks), calculated in the lower-level coordinated control module. Moreover, in our coordinated control and planning framework, we explicitly reason about the number of targets assigned to an action agent based on the respective action agent's dynamics and motion restrictions, such as maximum velocity. The effective performance of our perception-action collaboration and the described framework is empirically validated in our demonstrations and physical-robot experiments (presented in subsection 5.2.6).

Note: We assume that agents do not fail (e.g., UAV failure) and that the underlying motion model of the targets is known. In order to account for model deviations, we leverage an EKF-based system to estimate the parameters of this model, resulting in a measurement uncertainty based analytical TEB, introduced in Equation 5.42. Accordingly, three parameters, the process and observation noise covariances, Q_t and Γ_t, as well as the $\mathscr{E}$ in TEB, can be used to tune the reliability of the utilized motion model. If the assumed target motion model is accurate but the UAV sensors are noisy, Q_t and Γ_t can be initialized with small and large values to put more weight on the model and less on the UAV observation model, respectively (and vice versa). Moreover, the $\mathscr{E}$ in TEB can be tuned to more conservative values (e.g., larger values) for when the underlying process and observation models are less accurate, and vice versa.

In our low-level coordinated control framework we leveraged the CE-TSP for perception agents to account for their FOV; however, we note that it is an application-specific consideration to decide whether the Action agents can actually actuate at a certain distance with respect to targets. While in some applications the exact location of a moving target may be required, in many other cases such as our wildfire fighting case-study, oil-spill control in oceans or search-and-rescue, there exist numerous static or dynamic targets where action agents can actuate (e.g., dump water extinguisher on fire) *close-enough* to target locations. In such cases, as discussed in paragraph 5.2.4, our EKF-based system provides the MMSE estimate of the posterior which is assumed to be acceptable. Moreover, as noted above, the hyper-parameter $\mathscr{E}$ in Equation 5.42 can also be tuned based on the background application to improve the tracking accuracy to a satisfying threshold. Nevertheless, in our framework, the accuracy of the action agents moving to the exact target locations remains dependent on the model and measurement accuracy as well as well-tuning Equation 5.42.

5.2.8 Conclusion

In this paper, we have introduced a novel hierarchical approach to tackle the high-level decision making and low-level collaborative control problems for heterogeneous teams of autonomous robots consisting of perception agents and action agents. In our centralized high-level decision-making module, we propose MA-SARTSA-based learning under our MA-POSMDP model to enable perception agents to explore an unknown environment (i.e., discover dynamic targets) and exploit known targets by extracting their state information. Extracted state information is required for action agents for manipulation.

We have also introduced a measurement-uncertainty based tracking error and derived a set of analytical upper-bound service times to ensure a probabilistically guaranteed service for action agents in various scenarios. Additionally, we introduced a coordinated routing problem with an attribute-based robot-interaction scheme through which the perception-action agents cooperation is individualized to account for robots heterogeneity and improve the composite team's resiliency and performance.

CHAPTER 6
LEARNING END-TO-END MULTI-AGENT COORDINATION AND COMMUNICATION POLICIES

In previous chapters we mostly studied model-based and heuristic approaches for designing coordinated control and planning frameworks for collaborative robot teams. Although the agents in a multi-robot system can be programmed with hand-engineered heuristics and behaviors designed in advance to coordinate and communicate (as in section 5.1 and section 5.2), it is preferred that the agents can learn such new emergent behaviors from scratch to enable adaptability to complexity and changes in the environment [237].

In this chapter, we provide several contributions towards learning multi-agent collaborative policies. In particular, we study the application of MARL for learning communicative heterogeneous policies for a team of collaborating robots. The work presented in section 6.1 introduces a heterogeneous Multi-Agent Reinforcement Learning (MARL) architecture for learning highly efficient diverse communication among agents of a composite team [13, 10, 238].

6.1 Learning Efficient Communication for Cooperative Heterogeneous Teaming

High-performing teams learn intelligent and efficient communication and coordination strategies to maximize their joint utility. These teams implicitly understand the different roles of heterogeneous team members and adapt their communication protocols accordingly. MARL seeks to develop computational methods for synthesizing such coordination strategies, but formulating models for heterogeneous teams with different state, action, and observation spaces has remained an open problem. Without properly modeling agent heterogeneity, as in prior MARL work that leverages homogeneous graph networks, communication becomes less helpful and can even deteriorate the cooperativity and team performance. We propose

Heterogeneous Policy Networks (HetNet) to learn efficient and diverse communication models for coordinating cooperative heterogeneous teams. Building on heterogeneous graph-attention networks, we show that HetNet not only facilitates learning heterogeneous collaborative policies per existing agent-class but also enables end-to-end training for learning highly efficient binarized messaging. Our empirical evaluation shows that HetNet sets a new state of the art in learning coordination and communication strategies for heterogeneous multi-agent teams by achieving an 8.1% to 434.7% performance improvement over the next-best baseline across multiple domains while simultaneously achieving a 200× reduction in the required communication bandwidth.

6.1.1 Introduction and Motivation

High-performing human teams benefit from communication to build and maintain shared mental models to improve team effectiveness [239, 240]. Information sharing is key in building team cognition, and enables teammates to cooperate to successfully achieve shared goals [241, 12, 240]. Typical communication patterns across human teams widely differ based on the task or role the human assumes [242]. The field of MARL [243] has sought to develop agents that autonomously learn coordination and communication strategies to emulate high-performing human-human teams [244, 245, 246, 247, 248]. Yet, these approaches have fallen short in properly modeling heterogeneity and communication overhead in teaming [75, 67, 249, 250].

Heterogeneity in robots' design characteristics and their roles are introduced to leverage the relative merits of different agents and their capabilities [52, 10, 9, 8] We define a heterogeneous robot team as a group of cooperative agents that are capable of performing different tasks and may have access to different sensory information. We categorize agents with similar state, action, and observation spaces in the same *class*. In such a heterogeneous setting, communicating is not straightforward as agents do not speak the same "language"; we consider scenarios in which agents have different action-spaces and observation inputs from the environment (i.e., due to different sensors) or may not even have access to any observation input (i.e., lack of sensors, broken or low-quality sensors). The dependency generated via sensor-lax or sensor-void agents on agents with strong sensing capabilities makes efficient communication protocols for cooperation a requirement rather than an additional modeling technique for performance improvement.

While MARL researchers have increasingly focused on developing computational models of team communication [75, 82], most of these prior frameworks fail to explicitly model the heterogeneity of *composite teams* and fail to explicitly quantify and reduce the team's communication overhead to support decentralized, bandwidth-limited teaming. In this work, we intend to push the boundaries beyond this goal and seek to significantly reduce the bandwidth needs for communication to minimize communication overhead and facilitate practical implementation of our framework by designing a *decentralized* execution paradigm.

Contributions

Inspired by heterogeneous communication patterns across human teams, we propose HetNet to learn efficient and diverse communication models for coordinating cooperative heterogeneous robot teams. The key to our approach is the design of an end-to-end communication learning model with a differentiable encoder-decoder channel to account for the heterogeneity of inter-class messages, "translating" the encoded messages into a shared, intermediate language among agents of a composite team. Our empirical validation shows that HetNet's novel graph-based architecture achieves a new SOTA in learning emergent cooperative behaviors in complex, heterogeneous domains. HetNet achieves this result while also reducing communication overhead through intelligent message binarization, compressing the number of communicated bits needed by more than 200$\times$ per round of communication over the best performing baseline. **Contributions:**

1. We develop a novel, end-to-end heterogeneous graph-attention architecture for MARL that facilitates learning efficient, heterogeneous communication protocols among

cooperating agents to accomplish a shared task.

2. We design a differentiable encoder-decoder communication channel to learn efficient binary representations of states as an intermediate language among agents of different types to improve their cooperativity. Our binarized communication model achieves $200\times$ reduction in the number of communicated bits per round of communication over baselines while also setting a new SOTA in team performance.

3. We develop Multi-Agent Heterogeneous Actor-Critic (MAHAC) to learn *class-wise* cooperation policies in composite robot teams. Our results show the per-class critic structure achieves better performance over a centralized critic while having fewer model parameters than a per-agent critic.

4. We present empirical evidence that show HetNet is robust to varying bandwidth limitations and team compositions, setting a new SOTA in learning emergent cooperative policies by achieving at least an 8.1% to 434.7% performance improvement over baselines and across domains.

6.1.2 Related Work

MARL with Communication – Recently, the use of communication in MARL has been shown to enhance the collective performance of learning agents in cooperative MARL problems [251, 75, 67, 68, 76, 253, 244, 254, 255, 252, 5]. DIAL [66] and CommNet [68] displayed the capability to learn a discrete and continuous communication vectors, respectively. While DIAL considers the limited-bandwidth problem, neither of these approaches are readily applicable to composite teams or capable of performing attentional communication. TarMAC [67] achieves targeted communication through an attention mechanism which improves performance compared to prior work. Nevertheless, TarMAC requires high-bandwidth message passing channels and its architecture is reported to perform poorly in capturing the topology of interaction [76]. SchedNet [69] explicitly addresses the bandwidth-related concerns. However, in SchedNet agents learn how to schedule themselves for accessing the communication channel, rather than learning the communication protocols from scratch. In our approach, we explicitly address the heterogeneous communication problem where agents learn diverse communication protocols and intermediate language representations to use among themselves for cooperation. Our model enables agents to perform attentional communication and sending limited-length digitized messages through class-specific encoder-decoder channels, addressing the limited-bandwidth issues.

MARL with Graph Neural Network (GNN) – Prior work on MARL have sought to utilize GNNs to model a communication structure among agents [80]. Deep Graph Network (DGN) [79] represents dynamic multi-agent interaction as a graph convolution to learn cooperative behaviors. This seminal work in MARL demonstrates that a graph-based representation substantially improves performance. In [81], an effective communication topology is proposed by using hierarchical GNNs to propagate messages among groups and agents. G2ANet [76] proposed a game abstraction method combining a hard and a soft-attention mechanism to dynamically learn interactions between agents. More recently, MAGIC [82, 256] introduced a scalable, attentional communication model for learning a centralized scheduler to determine when to communicate and how to process messages through graph-attention networks. While these prior work have successfully modeled multi-agent interactions, they are not designed to address heterogeneous teams directly. HetNet, on the other hand, is designed to capture the heterogeneity among agents and learn an efficient shared language across agents with different action and observation spaces to improve cooperativity.

Heterogeneity in Multi-agent Systems – In [85], several types of heterogeneity induced by agents of different capabilities are discussed. As opposed to homogeneous teams, the diversity among agents in heterogeneous teams makes it challenging to hand-design intelligent communication protocols [85]. In [86], a control scheme is hand-designed for a heterogeneous multi-agent system by modeling the interaction as a leader-follower system. More recently, HMAGQ-Net [87] utilized GNNs and Deep Deterministic Q-network (DDQN) to facilitate coordination among heterogeneous agents (i.e., those with different state and action spaces). Going beyond this prior work, we build our HetNet model based upon an actor-critic framework and generalize the problem formulation for state-, action- and observation-space heterogeneities. Moreover, HetNet facilitates learning efficient binary representations of states as an intermediate language among agents of different types to improve cooperativity. To the best of our knowledge, we are the first to generate a general multi-agent coordination framework for heterogeneous robot teams that is able to learn a communication protocol for high-performance coordination.

6.1.3 Preliminaries

Problem Formulation

Founding on a standard POMDP [257], we formulate a new problem setup termed as Multi-Agent Heterogeneous Partially Observable Markov Decision Process (MAH-POMDP), which can be represented by a 9-tuple $\langle \mathscr{C}, \mathscr{N}, \{\mathscr{S}^{(i)}\}_{i \in \mathscr{C}}, \{\mathscr{A}^{(i)}\}_{i \in \mathscr{C}}, \{\Omega^{(i)}\}_{i \in \mathscr{C}}, \{\mathscr{O}^{i}\}_{i \in \mathscr{C}}, r, \mathscr{T}, \gamma \rangle$. $\mathscr{C}$ is set of all available agent classes in the composite robot team and the index $i \in \mathscr{C}$ shows the agent class. $\mathscr{N} = \sum_{\langle i \in \mathscr{C} \rangle} N^{(i)}$ is the total number of collaborating agents where $N^{(i)}$ represents the number of agents in class i. $\{\mathscr{S}^{(i)}\}_{i \in \mathscr{C}}$ is a discrete joint set of state-spaces. For each class-dependent state-space, $\mathscr{S}^{(i)}$, we have $\mathscr{S}^{(i)} = \left[s_t^{i_1}, s_t^{i_2}, \cdots, s_t^{i_{N^{(i)}}} \right]$, where $s_t^{i_j}$ represents the state-vector of agent j of the i-th class, at time t. $\{\mathscr{A}^{(i)}\}_{i \in \mathscr{C}}$, is a discrete joint set of action-spaces. For each state-dependent action-space, $\mathscr{A}^{(i)}$, we have $\mathscr{A}^{(i)} = \left[a_t^{i_1}, a_t^{i_2}, \cdots, a_t^{i_{N^{(i)}}} \right]$, forming the joint action-vector of agents of class i at time t. $\{\Omega^{(i)}\}_{i \in \mathscr{C}}$ is similarly defined as the joint set of observation-spaces, including class-specific observations. $\gamma \in [0, 1)$ is the temporal discount factor for each unit of time and $\mathscr{T}$ is the state transition probability density function.

At each timestep, t, each agent, j, of the i-th class can receive (if the observation input is enabled for class i) a partial observation $o_t^{i_j} \in \Omega^{(i)}$ according to some class-specific observation function $\{\mathscr{O}^{(i)}\}_{i\in\mathscr{C}} : o_t^{i_j} \sim \mathscr{O}^{(i)}(\cdot|\bar{s})$. If the environment observation is not available for agents of class i, agents in the respective class will not receive any input from the environment (e.g., lack of sensory inputs). Regardless of receiving an observation or not, at each time, t, each agent, j, of class i, takes an action, $a_t^{i_j}$, forming a joint action vector $\bar{a} = \left(a_t^{1_1}, a_t^{1_2}, \cdots, a_t^{i_1}, \cdots, a_t^{i_j}\right)$. When agents take the joint action $\bar{a}$, in the joint state $\bar{s}$ and depending on the next joint-state, they receive an immediate reward, $r(\bar{s},\bar{a}) \in \mathbb{R}$, shared by all agents and regardless of their classes. Our objective is to learn optimal policies per existing agent-class to solve the MAH-POMDP by maximizing the total expected, discounted reward accumulated by agents over an infinite horizon, i.e., $\arg\max_{\pi(\bar{s})\in\Pi} \mathbb{E}_{\pi(\bar{s})}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | \pi(\bar{s})\right]$.

Actor-Critic (AC) Methods

Actor-Critic (AC) methods [258, 259] are an approach to RL that utilize function approximation, in which each agent j has a policy, $\pi_\theta^j(a|s)$, parameterized by θ, that specifies which action, a, to take in each state, s, to maximize the expected future discounted reward. Actor-Critic (AC) methods apply gradient ascent to the actor's parameters, θ, based upon a *critic*, $Q^\phi(s,a)$, action-value function [260], parameterized by ϕ, where $Q^\phi(s,a)$ approximately solves the credit-assignment problem [261]. By the policy gradient theorem [126], the expected reward maximization (i.e., the AC objective), $J(\theta)$, is maximized via $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta^j}\left[\nabla_\theta \log \pi_\theta^j(a_t^j|o_t^j) Q^\phi(o_t^j, a_t^j)\right]$, where a_t^j and o_t^j are the action and observation of agent j, respectively.

Graph Neural Networks (GNNs)

GNNs are a class of deep neural networks that capture the structural dependency among nodes of a graph via message-passing between the nodes, where each node aggregates feature vectors of its neighbors to compute a new feature vector [262, 78, 79]. The canonical feature update procedure via graph convolution operator can be shown as $\bar{h}'_j = \sigma\left(\sum_{k\in N(j)} \frac{1}{c_{jk}} \omega \bar{h}_k\right)$, where $\bar{h}'_j$ is the updated feature vector for node j, $\sigma(.)$ is the activation function and, ω

represents the learnable weights. $k \in N(j)$ includes the immediate neighbors of node j where k is the index of neighbor, and c_{jk} is the normalization term which depends on the graph structure. A common choice of c_{jk} is $\sqrt{|N(j)N(k)|}$. In an L-layer aggregation, a node j's representation captures the structural information within the nodes that are reachable from j in L hops or fewer. However, the fact that c_{jk} is structure-dependent can impair generalizability of GNNs when scaling the graph's size. Thus, a direct improvement is to replace c_{jk} with attention coefficients, α_{jk}, computed via Equation 6.1. In Equation 6.1, $\bar{W}_{att}$ is the learnable weight, $\|$ represents concatenation, and $\sigma'(.)$ is the Leaky-ReLU nonlinearity. The Softmax function is used to normalize the coefficients across all neighbors k, enabling feature dependent and structure free normalization [263, 77].

$$\alpha_{jk} = \text{softmax}_k \left(\sigma' \left(\bar{W}_{att}^T \left[\omega \bar{h}_j \, \| \, \omega \bar{h}_k \right] \right) \right) \tag{6.1}$$

6.1.4 Method

In this section, we first present an overview of the communication problems and constraints considered in our work. We then describe how to construct a heterogeneous graph given a problem state and present the building block layer, which we refer to as Heterogeneous Graph-Attention (HetGAT) layer, and develop a binarized encoder-decoder communication channel to account for the heterogeneity of messages passed among agents. Eventually, we cover the logistics of utilizing HetGAT layers to assemble our heterogeneous policy network, HetNet, of arbitrary depth.

Communication Problem Overview

In this work, we are concerned with the problem of coordinating a robot team via fostering direct communication among interacting agents. We consider MARL problems wherein multiple agents interact in a single environment to accomplish a task which is of a cooperative nature. We are particularly interested in scenarios in which the agents are heterogeneous in

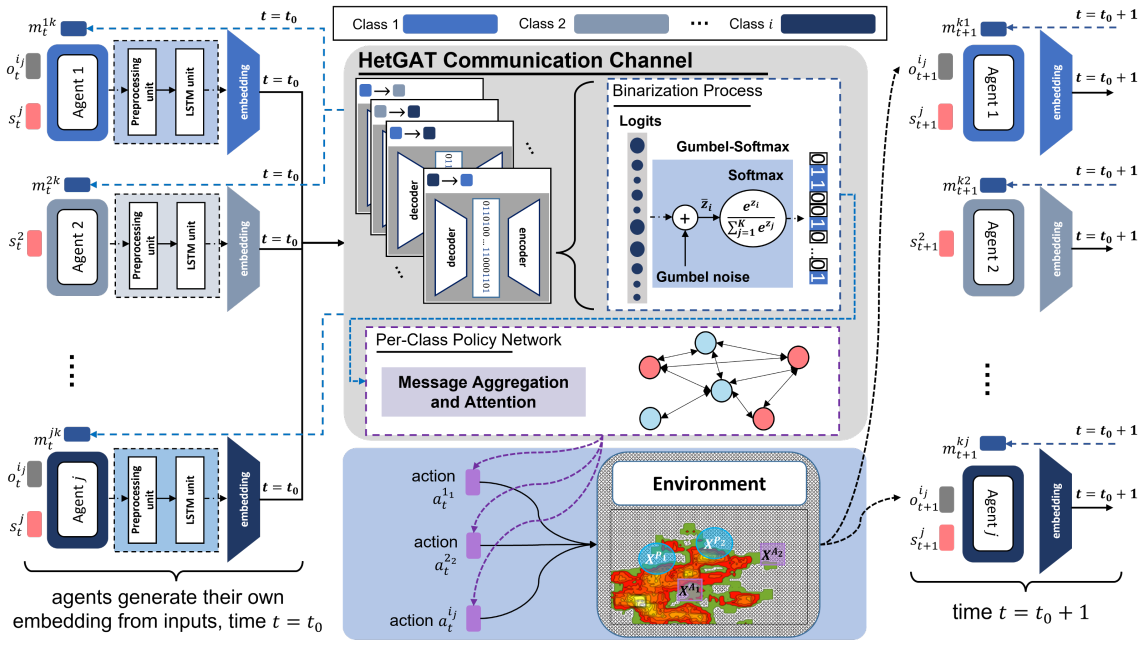


Figure 6.1: Overview of our multi-agent heterogeneous attentional communication architecture in a Centralized Training and Decentralized Execution (CTDE) paradigm. At each time point $t = t_0$, each agent j of class i generates a local embedding from its own inputs, by passing its input data through class-specific preprocessing units (i.e., a Convolutional Neural Network (CNN) or a fully-connected NN) and a Long Short-Term Memory (LSTM) cell. The embeddings are then passed into the HetGAT communication channels by each agent where a Gumbel-Softmax process is used in a class-specific encoder network to convert the message into binarized messages, m_t^{jk},. The message is then sent to neighboring agents where a class-specific decoder is used to decode all the received messages, aggregate them and weight them according to some learned attention coefficients. The aggregated message information is leveraged by the receiving agent to compute the action probabilities as its policy output. These decentralized per-class policies are then used for action decisions of different agents in each class.

their capabilities, meaning agents can have different state, action and observation spaces in forming a composite team.

In learning an end-to-end communication model, we take a series of problems and constraints into consideration: (1) heterogeneous messages, where agents of different classes have different action and observation spaces, resulting in different interpretations of sent/received messages; (2) Attentional and scalable communication protocols, such that agents incorporate attention coefficients depending on the agent/class they are communicating with for coordinating with teammates in any arbitrary team sizes; (3) Learning communication models for Low-Size, -Weight, and -Power (Low-SWAP) systems, where due to limited communication bandwidth, agents must learn to communicate in a highly efficient shared intermediate "language" (e.g., limited-length binarized messages); (4) Limited-range communications, where agents can only exchange messages when they are within a close proximity.

Heterogeneous Communication Model

GNNs previously used in MARL operate on homogeneous graphs to learn a universal feature update and communication scheme for all agents [79, 81, 76, 82], which fails to explicitly model the heterogeneity among agents. We instead cast the cooperative MARL problem into a heterogeneous graph structure, and propose a novel heterogeneous graph-attention network capable of learning diverse communication strategies based on agent classes. Compared to homogeneous graphs, a heterogeneous graph can have nodes and edges of different types that can have different types of attributes. This advantage greatly increases a graph's expressivity and enables straightforward modeling of complicated, composite teams.

Given our MAH-POMDP formulation in subsubsection 6.1.3, we directly model each agent class $i \in \mathscr{C}$ as a unique node type. This approach allows agents to have different types of state-space content, $\mathscr{S}^{(i)}$, as input features according to their classes, as well as enabling different types of action spaces, $\mathscr{A}^{(i)}$. Communication channels between agents are modeled as directed edges connecting the corresponding agent nodes. When two agents move to a close proximity of each other such that those agents fall within communication range, we add bidirectional edges to allow message passing between them. We use different edge types to model different class combinations of the sender and receiver agents to allow for learning heterogeneous communication protocols and intermediate representations.

To form our novel architecture for modeling heterogeneous interactions, we add a State Embedding Node (SEN) into the heterogeneous graph to train a critic network. SEN serves as a central node where we aggregate all the important meta-data from the MARL environment (i.e., number of agents, $\mathscr{N}$, world size, current time step, etc.) and use the embeddings for critic training. The SEN forms a *one-way* connection to the agent nodes (i.e., from an agent to the SEN) to receive messages from them during training. The SEN's learned embeddings are used as input of a critic network consisting of one Fully-Connected (FC) layer for state-dependent value estimation. We note that, since there are no edges pointing from the SEN to any agent nodes, during the execution phase, the SEN can be safely removed without affecting an agent's own policy output, which complies with our underlying CTDE paradigm.

Accordingly, we present an overview of our multi-agent heterogeneous attentional communication architecture in Figure 6.1. At each time, t, the features of each agent (i.e., each node of the heterogeneous graph) are generated through a class-specific feature preprocessor. We utilize separate modules to preprocess an agent's state-vector, $s_t^{i_j}$, and observation, $o_t^{i_j}$, since depending on the agent's class, the environment observation input may not be available (e.g., an action agent in a perception-action composite team). Each preprocecssing module contains one CNN or a fully-connected unit followed by one LSTM cell to enable reasoning about temporal information. As shown in Figure 6.1, the generated embeddings are then passed into a HetGAT communication channel including a class-specific encoder-decoder network and a Gumbel-Softmax [264] unit to generate a binarized message, m_t, for an agent, j.

Binarized Communication Channels

The feature update process in a HetGAT layer is conducted in two steps: *per-edge-type* message passing followed by *per-node-type* feature reduction. When modeling multi-agent teams, we reformulate the computation process into two phases: a *sender* phase and a *receiver* phase. Figure 6.2 shows the computation flow during the sender and receiver phases for an agent, j, of class i.

During the sender phase, the agent, j, of class $i \in \mathscr{C}$, processes its input feature, h_j, using a class-specific weight matrix, $\omega_i \in \mathbb{R}^{d' \times d}$, where d and d' are the input and output feature dimensions, respectively. The agent also transforms h_j into the assigned message dimension using a class-specific encoder, $\omega_i^{enc} \in \mathbb{R}^{n \times d}$, where n is the communication channel bandwidth. Next, we leverage a universal binarization process utilizing Gumbel-Softmax to convert the message into 0s and 1s for all classes as an efficient, intermediate language. The binarized message is then sent to neighboring agents.

During the receiver phase, agent, j, of class i, processes all the received messages using a class-specific decoder, $\omega_i^{dec} \in \mathbb{R}^{d' \times n}$. Next, for each type of the communication edge that an agent is connected to, the HetGAT layer computes per-edge-type aggregation result by weighing received messages, along the same edge-type with normalized attention coefficients, $\alpha^{edgeType}$. The aggregation results are then merged with the agent's own transformed embedding, $\omega_i h_j$, to compute the output features. The feature update formula for an agent is shown in Equation 6.2, where j and k are agent indexes and, $i, l \in \mathscr{C}$ are class indexes; such that, $i2l$ is an *edgeType* and means "from class i to class l". m_t^{jk} is the decoded message computed by Equation 6.3 and, $\Delta_l(j)$ include agent j's neighbors that belong to class l.

$$\text{Class } (i): \quad \bar{h}'_j = \sigma\left(\omega_i \bar{h}_j + \sum_{l \in \mathscr{C}} \sum_{k \in N_l(j)} \alpha_{jk}^{i2l} m_t^{jk}\right) \tag{6.2}$$

$$m_t^{jk} = \omega_i^{dec}(\text{GumbelSoftmax}(\omega_l^{enc} h_k)) \tag{6.3}$$

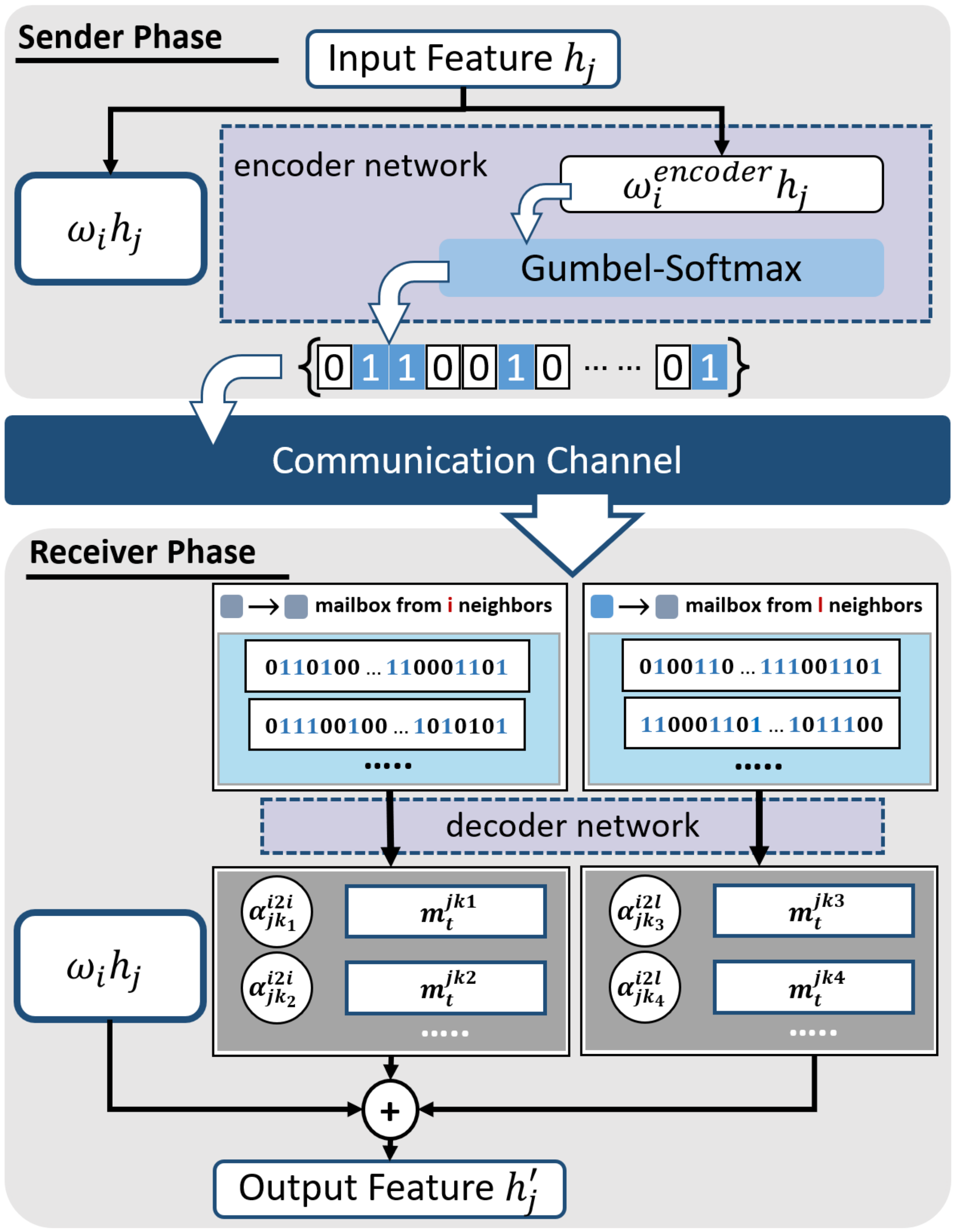


Figure 6.2: The sender and receiver phases of the feature update process in a HetGAT layer for one agent, j, of class i.

Note that we have $l = i$ for intra-class communication. When computing attention coefficients in a heterogeneous graph, we adapt Equation 6.1 into Equation 6.4 to account for heterogeneous channels.

$$\alpha_{jk}^{i2l} = \text{softmax}_k \left(\sigma' \left(\bar{W}_{att}^T \left[\omega_i \bar{h}_j \,\|\, \omega_{i2l} \bar{h}_k \right] \right) \right) \tag{6.4}$$

As discussed in subsubsection 6.1.4, we add an SEN to the graph during centralized training with a state-dependent critic network. The feature update formula of the SEN is shown in Equation 6.5. Here, feature vectors from all agents are passed to the SEN after being processed with edge-specific weights, $\omega_{edgeType}$. The attention coefficients for the SEN are computed in a similar manner as in Equation 6.4.

$$\text{SEN}: \quad \bar{h}_s' = \sigma \left(\omega_s \bar{h}_s + \sum_{i \in \mathscr{C}} \sum_{j \in N^{(i)}} \alpha_{sj}^{i2sen} \omega_{i2sen} \bar{h}_j \right) \tag{6.5}$$

To stabilize the learning process, we adapt the multi-head extension of the attention mechanism [263] to fit our heterogeneous setting. We use L independent HetGAT (sub-)layers to compute node features in parallel and then merge the results by concatenation operation for each multi-head sub-layer in HetNet except for the last layer which employs averaging. As a result, each type of communication channel is split into L independent, parallel sub-channels.

Heterogeneous Policy Network (HetNet)

At each timestep, t, a HetGAT layer corresponds to one round of message exchange between neighboring agents and feature update within each agent. By stacking several HetGAT layers, we construct the HetNet model that utilizes multi-round communication to extract high-level embeddings of each agent for decision-making. For the last HetGAT layer in HetNet, we set each agent's output feature dimension to the size of its action-space, specific to its class, i. Then, for each agent node, we add a Softmax layer on top of its output to obtain a probability distribution over actions that can be used for action sampling, resulting in class-wise stochastic policies. Accordingly, the computation process of each agent's policy remains local for distributed execution, and the SEN is no longer needed.

6.1.5 Training and Execution

Multi-agent Heterogeneous Actor-Critic

We present our modified Multi-Agent Heterogeneous Actor-Critic (MAHAC) framework for learning class-wise coordination policies. We assign one policy per existing class, $\pi^i \in \{\Pi\}^{\mathscr{C}}$, each of which is parametrized by θ^i. The trained actor network on the heterogeneous graph contains one set of learnable weights per agent class, which due to the message-passing nature of GNN updates, can be distributed to individual agents in the execution phase. Accordingly, in MAHAC, the policy for each class, π^i, is updated by a variant of the basic AC objective (see subsubsection 6.1.3, shown in Equation 6.6. We leverage an on-policy training paradigm for MAHAC.

$$\nabla_{\theta^i} J(\theta^i) = \frac{1}{N} \sum_{j=1}^{\mathscr{N}^{(i)}} \sum_{t=1}^{T} \nabla_{\theta^i} \log \pi^i \left(\bar{a}_t^{i_j} | \bar{o}_t^{i_j}, \bar{m}_t \right) \left(\left(\sum_{t'=t}^{T} \gamma^{t'-t} r^{i_j}(\bar{s}^{i_j}, \bar{a}^{i_j}) \right) - b(t) \right) \tag{6.6}$$

In Equation 6.6, $\bar{a}_t^{i_j}$ and $\bar{o}_t^{i_j}$ represent the joint actions taken and joint observations received (if applicable for class i) by agents at time, t. $\bar{m}_t$ represents the message-vector received by agent j from its neighbors. The term $\sum_{t'=t}^{T} \gamma^{t'-t} r^{i_j}(\bar{s}^{i_j}, \bar{a}^{i_j})$ calculates the total discounted future reward from current timestep to end of an episode. Moreover, $b(t)$ is a temporal baseline function leveraged to reduce the variance of the gradient updates in MAHAC. We utilize the value-estimates via our critic network as the baseline function [265][1].

Critic Architecture Design for HetNet

In this section, we propose and assess several MAHAC architectures to investigate the utility of: (1) *fully-centralized* critic, $b(t)$ (i.e., one critic signal for all), (2) *per-class* critics, $b^i(t)$ (i.e., one critic signal per class of agents) and (3) *per-agent* critics, $b^{i_j}(t)$ (i.e., individual critic signals for each agent) to learn class-wise policies.

[1]We provide our code at https://github.com/CORE-Robotics-Lab/HetNet

In the *fully-centralized critic* implementation for HetNet, we stack an FC layer on top of SEN's output feature for critic prediction of the value estimate. The same predicted critic value is used in the policy gradient update for all agents of all classes. The target value for training the critic output is the average returns (i.e., discounted sum of future rewards) over all agents. Thus, in this architecture one centralized critic network "criticizes" the actions of all agents. Note that this approach still complies with our CTDE paradigm, since the actor network is implemented on a GNN structure.

For the *per-class critic* implementation, we split the critic head into one critic head per existing agent class to separate the critic estimation for different types of agents. The critic split is done while the critic is estimated based on a class-specific SEN's output feature. During training, the target value for each class of critic output is the average returns over the same class of agents. Algorithm algorithm 5 provides a pseudocode to train HetNet with the per-class critic architecture.

In our *per-agent critic* implementation for HetNet, the critic network outputs one critic value for each agent. This is achieved by concatenating the SEN's output feature with each agent node's output embedding to serve as the input of class-specific critic heads. The per-agent critic estimation is used for each agent's policy update where the target value for training is the returns of that agent.

6.1.6 Empirical Evaluation

Evaluation Environments

We evaluate the utility of HetNet against several baselines in three cooperative MARL domains (a homogeneous and two heterogeneous) that require learning collaborative behaviors. Please refer to Appendix C for environment and model details.

Predator-Prey (PP) [251] – For the homogeneous domain, we adopt the Predator-Prey (PP) [251] in which the goal is for *N* predator agents with limited vision to find a stationary prey and move to its location. The agents in this domain all belong to the same class (i.e.,

Algorithm 5: The *Per-class* training procedure for HetNet.

```
1: Input: Agent classes, $i \in \mathscr{C}$, number of agents in each class, $\mathscr{N}^{(i)}$, number of episodes
   per epoch $K$, maximum allowed steps for each episode, $T$, learning rate, $\eta$.
2: Initialize: Per-class policy parameters $\{\theta^i\}$ for $\{\pi^i\}$ and per-class critic parameters
   $\{\phi^i\}$ for $\{V^i\}$, $i \in \mathscr{C}$
3: while not converged do
4:    Sample a random environment instance
5:    for $k = 1$ to $K$ do
6:       Get initial observations $\{o_1^{1_1}, o_1^{1_2}, ..., o_1^{i_j}\}$, $i \in \mathscr{C}$, $j \in \mathscr{N}^{(i)}$
7:       for $t = 1$ to $T$ do
8:          Perform message passing and feature reduction
9:          Store critic predictions $\{V_t^i\}$, $i \in \mathscr{C}$
10:         Sample actions: $a_t^{i_j} \sim \pi^i(* \mid o_t^{i_j})$, $i \in \mathscr{C}$, $j \in \mathscr{N}^{(i)}$
11:         Step through environment using $\{a_t^{1_1}, a_t^{1_2}, \cdots, a_t^{i_j}\}$, receive next observations
            and rewards: $\{o_{t+1}^{1_1}, o_{t+1}^{1_2}, ..., o_{t+1}^{i_j}\}$, $\{r_t^{1_1}, r_t^{1_2}, ..., r_t^{i_j}\}$
12:         if environment_solved then: Terminate early end if
13:      end for
14:   end for
15:   for $i \in \mathscr{C}$ do
16:      Compute rewards-to-go $R_t^i$ and GAE advantages $A_t^i$
17:      $\nabla J(\theta^i) = \frac{1}{N} \sum_{j=1}^{\mathscr{N}^{(i)}} \sum_{t=1}^{T} \nabla \log \pi^i \left( a_t^{i_j} | o_t^{i_j} \right) A_t^i$
18:      Critic loss: $L(V^i) = \frac{1}{N} \sum_{j=1}^{\mathscr{N}^{(i)}} \sum_{t=1}^{T} \left( V_t^i - R_t^i \right)^2$
19:      Joint update: $\theta^i = \theta^i + \eta \nabla J(\theta^i)$, $\phi^i = \phi^i - \eta \nabla L(V^i)$
20:   end for
21: end while=0
```

identical state, observation and action spaces).

Predator-Capture-Prey (PCP) – For the first heterogeneous domain, we modify the PP to create a new environment, which we refer to as Predator-Capture-Prey (PCP), to include a composite team. In PCP, we have two classes of *predator* and *capture* agents. Agents of the *predator* class have the goal of finding the prey with limited vision (similar to agents in PP). Agents of the *capture* class, have the goal of locating the prey *and* capturing it with an additional *capture-prey* action in their action-space, while not having any observation inputs (e.g., lack of scanning sensors).

FireCommander (FC) [54] – In the second heterogeneous domain, the FireCommander [54], two classes of *perception* and *action* agents must collaborate as a composite team to

Table 6.1: Reported results are Mean (± Standard Error (SE)) from 50 evaluation trials. For all tests, the final training policy at convergence is used for each method. As shown, HetNet outperforms all baselines in all three domains.

Method	PP		PCP		FC	
	Avg. Cumulative $\mathscr{R}$	Avg. Steps Taken	Avg. Cumulative $\mathscr{R}$	Avg. Steps Taken	Avg. Cumulative $\mathscr{R}$	Avg. Steps Taken
TarMAC [67]	-0.563 ± 0.030	18.4 ± 0.46	-0.548 ± 0.031	17.0 ± 0.80	-109.2 ± 6.26	248.1 ± 6.97
IC3Net [251]	-0.342 ± 0.015	9.69 ± 0.26	-0.411 ± 0.019	11.5 ± 0.37	-187.2 ± 0.79	276.0 ± 5.51
CommNet [68]	-0.336 ± 0.012	8.97 ± 0.25	-0.394 ± 0.019	11.3 ± 0.34	-253.2 ± 1.01	292.7 ± 3.07
MAGIC [82]	-0.386 ± 0.024	10.6 ± 0.50	-0.394 ± 0.017	10.8 ± 0.45	-267.6 ± 10.9	298.1 ± 23.3
HetNet [Ours]	**-0.232 ± 0.010**	**8.30 ± 0.25**	**-0.364 ± 0.017**	**9.98 ± 0.36**	**-9.862 ± 2.77**	**46.40 ± 2.90**

extinguish a propagating firespot. At each timestep, the firespot propagates to a new location according to the FARSITE [122] model, while the previous location is still on fire. All firespots are initially hidden to agents and need to be discovered before being extinguished. As such, *perception* agents are tasked to scan the environment to detect the firespots while *action* agents (no observation inputs) are required to move and extinguish a firespot that has been discovered by a *perception* agent before. Note that since firespots propagate, both *perception* and *action* agents need to continue to explore the map and collaborate until all firespots are extinguished.

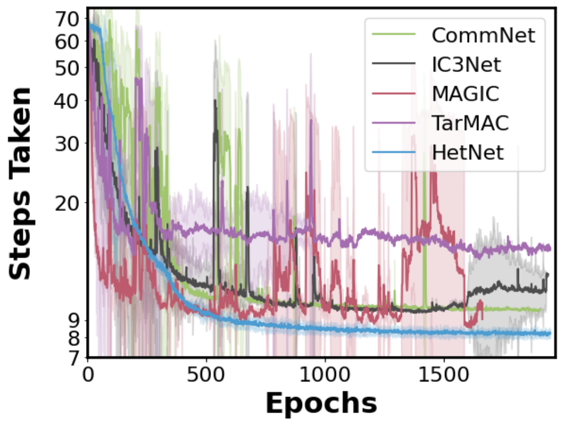
Baselines

We benchmark two variants of our framework, i.e. HetNet-Binary and HetNet-Real, against four end-to-end communicative MARL baselines: (1) CommNet [68], (2) IC3Net [251], (3) TarMAC [67] and, (4) MAGIC [82]. For our HetNet-Real variant, we remove the binarization process (i.e., Gumbel-Softmax) and the encoder-decoder network from the communication channel. Accordingly, agents directly send their generated embeddings (i.e., the LSTM cell output) to a class-specific communication edge in HetGAT layers. The HetNet-Real utilizes continuous, agent-specific embeddings to generate limited-length, real-valued numbers which allow for greater expressivity in the message-space. The real-valued numbers require more communication band-width and higher memory storage as compared to HetNet-Binary (see subsubsection 6.1.6). We note that, for all four baselines, i.e. CommNet [68], IC3Net [251], TarMAC [67] and, MAGIC [82], we directly pulled the respective authors' publicly available code-bases and hyperparameters for training. Note that we observed some performance discrepancies while directly using MAGIC's public repository (i.e., `github.com/MAGIC`).

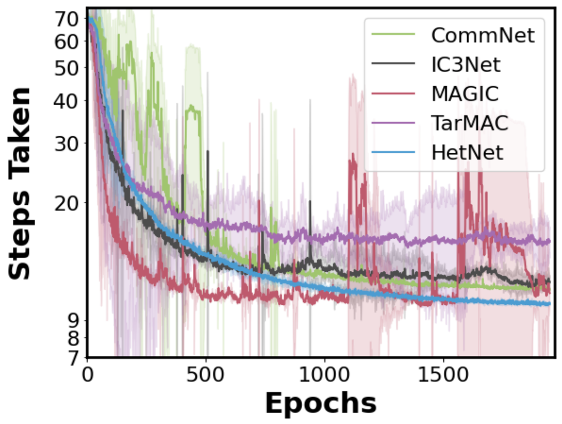
Results, Ablation Studies, and Discussion

Here, we empirically validate the performance of our frameworks, across homogeneous and heterogeneous teaming domains and against the introduced baselines. Next, we present an ablation study to investigate the required communication overhead for each method (paragraph 6.1.6). We then present evidence to support the effects of communication on collaboration performance (paragraph 6.1.6) as well as to determine the sensitivity of HetNet to key variables such as number of agents (paragraph 6.1.6). Additionally, we investigate the effects of the critic structures proposed in subsubsection 6.1.5 on HetNet's performance (paragraph 6.1.6).

Baseline Comparison Figure 6.3 depicts the average steps taken (± standard error) by each method across episodes as training proceeds in PP and PCP domains. In both domains, PP and PCP, HetNet outperforms all baselines by converging to a more efficient coordination policy (i.e., fewer steps taken). We also tested the learned coordination policies at convergence by each of the baselines in PP, PCP and FC domains. The results of this test are presented in Table 7.1 where the reported results are mean (± Standard Error (SE))



(a) Homogeneous Domain (PP)



(b) Heterogeneous Domain (PCP)

Figure 6.3: Average steps taken (± SE) by each method across episodes and three different random seeds as training proceeds. HetNet outperforms all baselines in both domains.

from 50 evaluation trials with different random-seed initializations. As shown, HetNet outperforms all baselines in all three domains. Additionally, in the same experiment, the coordination policy learned by our HetNet-Binary with 64-bits message dimensionality achieved 9.90±0.58 average steps taken in the PCP domain; showing better performance than all baselines while significantly compressing the communication bandwidth (see Figure 6.4). The heterogeneous policies learned by our model set the SOTA for learning challenging cooperative behaviors for composite teams.

Ablation Study #1: Communication Bandwidth In this experiment, we compute the Communication Bandwidth (CB) for each baseline as the number of bits required to communicate messages per round of communication during evaluation (i.e., converged policies deployed for test). As shown in Figure 6.4, HetNet facilitates binarized communication among agents which requires significantly less CB as compared to real-valued baselines (i.e., one bit per binary value vs. 64 bits in single-precision floating-point format [266]). HetNet-Binary with 64 and 32-bits messages, respectively, achieve more than 100× and 200× lower CB while showing better performance than real-valued baselines.

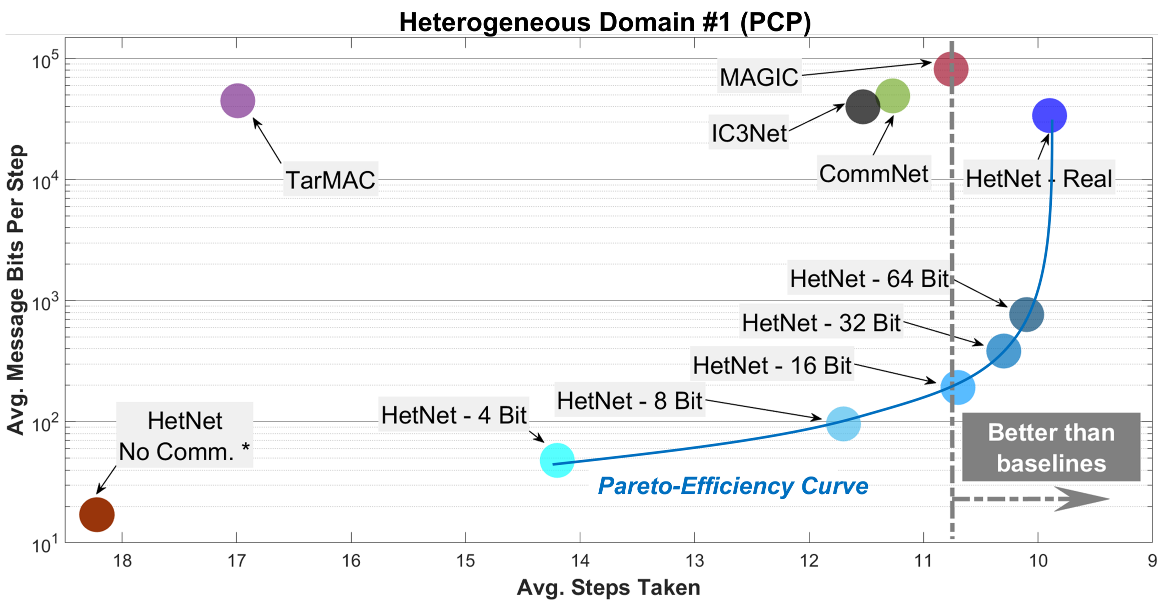
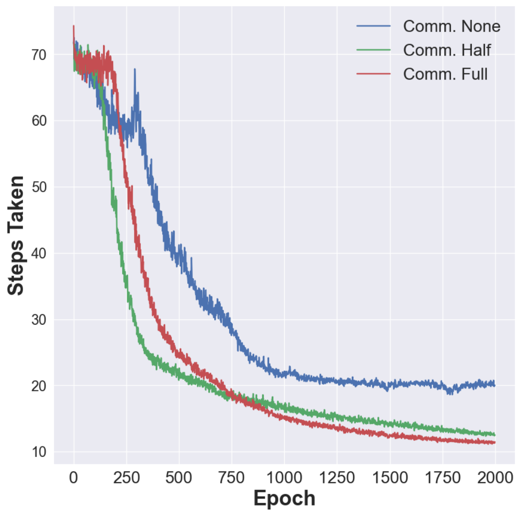


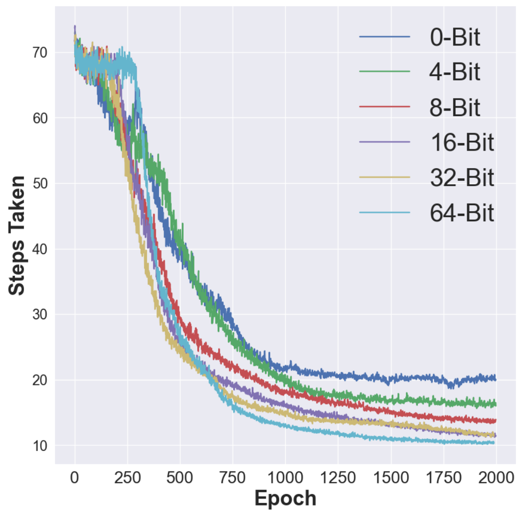
Figure 6.4: Communicated bits per round of communication vs. performance in PCP for different methods. HetNet facilitates binarized messages among agents which requires significantly less CB as compared to real-valued baselines.

Ablation Study #2: Effects of Communication We assess the impact of the communication on cooperation performance of the composite team. We present two experiments in the PCP domain for comparing HetNet's performance: (1) with *Full*, *Half* and *No* communication among agents and (2) with different binary message dimensions (number of bits). As depicted in Figure 6.5a, HetNet performs significantly better with full communication while the performance drop for half-communication (i.e., limited range) is not considerable. As such, the results show that our model, HetNet has robustness to degradation in communication range. Additionally, as shown in Figure 6.5b, a gradual degradation in performance is observed by decreasing message dimensionality rather than a sharp drop-off. HetNet's performance improves with longer messages as the learned intermediate language will have greater expressivity.

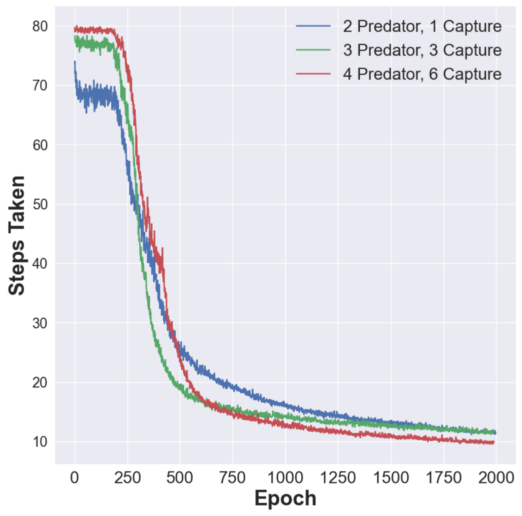
Ablation Study #3: Scalability to Number of Agents in the Composite Team In this experiment, we evaluated the scalability of our HetNet-Binary to different number of agents in the composite team. Specifically, we tested HetNet-Binary in PCP domain with (2P, 1C),



(a) Communication range.



(b) Message dimensionality.

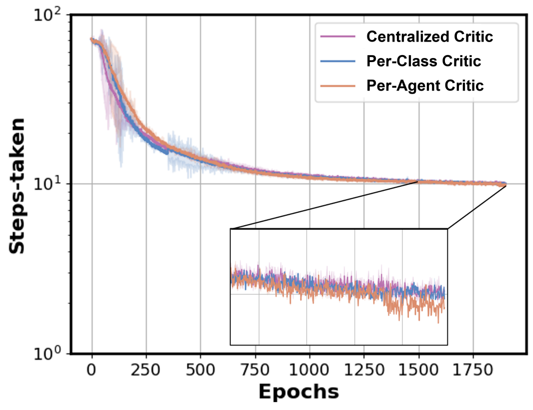


(c) Scalability to num. of agents.

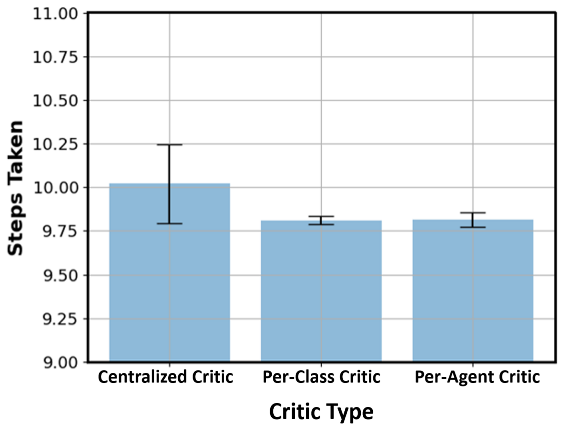
Figure 6.5: Analyzing HetNet's performance with and without communication (Figure 6.5a) and across different binary message dimensions (Figure 6.5b) in the PCP domain. Communication policy learned by HetNet improves the cooperativity among agents and the performance improves with larger message sizes. Figure 6.5c depicts results for analyzing HetNet's ability to scale to different number of agents. As shown, HetNet-Binary can successfully scale to different sizes of the composite team.

(3P, 3C) and (4P, 6C) team compositions, where *P* and *C* represent *predator* and *capture* agents, to evaluate the scalability to different team sizes. The results of this experiment are presented in Figure 6.5c. As shown, HetNet's GNN-based architecture can successfully scale to different combinations of the composite team by approximately converging at the same rates.

Ablation Study #4: Effects of the Critic Structures Finally, we investigate the utility and performance of the three critic structures proposed in subsubsection 6.1.5 on HetNet's performance in the PCP domain. We utilized our HetNet-Real variant for this experiment. Figure 6.6a shows the learning curves during training for centralized, per-class, and per-agent critic structures in the PCP domain. The test results for coordination policies learned by each of the critic architectures are presented in Figure 6.6b, showing the average number of steps taken to win the game by deploying the converged policies by each critic design. As depicted, HetNet-Real shows similar performance with per-class and per-agent critics, both having better results than the centralized critic, decreasing the number of steps of episode completion by 0.20 (10.01 $\rightarrow$ 9.81). The performance benefit can be attributed to the ability to utilize individual and class-wise rewards, both of which help to capture the heterogeneity in the received feedback from the environment.



(a) Training Performance.



(b) Converged Performance.

Figure 6.6: Learning curves during training as well as the test results (average number of steps taken) for final policies learned by centralized, per-class and per-agent critic architectures in the PCP domain.

6.1.7 Conclusion

Motivated by the diverse communication patterns across collaborating human teams, we present a communicative, cooperative MARL framework for learning heterogeneous cooperation policies among agents of a *composite* team. We propose Heterogeneous Policy

Network (HetNet), a heterogeneous graph-attention based architecture, and introduce the Multi-Agent Heterogeneous Actor-Critic (MAHAC) learning paradigm for training HetNet to learn class-wise cooperation policies. We push the boundaries beyond performance considerations as in prior work by equipping HetNet with a binarized encoder-decoder communication channel to facilitate learning a new and highly efficient encoded language for heterogeneous communication. We empirically show HetNet's superior performance against several baselines in learning both homogeneous and heterogeneous cooperative policies. We provide empirical evidence that show: (1) our binarized model achieves more than 200$\times$ reduction in communication overhead (i.e., message bits) per round of communication while also outperforming baselines in performance, (2) HetNet is robust to varying bandwidth limitations and team compositions.

CHAPTER 7
ITERATIVE REASONING FOR MULTI-AGENT DECISION-MAKING UNDER UNCERTAINTY

In addition to communication, individuals in high-performing human teams also benefit from the theory of mind and making strategic decisions by recursively reasoning about the actions (strategies) of other human members. Such hierarchical rationalization alongside with communication facilitate meaningful and strategic cooperation in human teams.

Inspired by this behavior in strategic human teams, in this chapter (section 7.1), we propose a novel information-theoretic, fully-decentralized cooperative MARL framework, called Informational Policy Gradient (InfoPG) [12], where agents iteratively rationalize their action-decisions based on their teammates' actions. We study cooperative MARL under the assumption of bounded rational agents and leverage action-conditional policies into policy gradient objective to accommodate our assumption.

7.1 Iterated Reasoning with Mutual Information in Cooperative and Byzantine Decentralized Teaming

Information sharing is key in building team cognition and enables coordination and cooperation. High-performing human teams also benefit from acting strategically with hierarchical levels of iterated communication and rationalizability, meaning a human agent can reason about the actions of their teammates in their decision-making. Yet, the majority of prior work in MARL does not support iterated rationalizability and only encourage inter-agent communication, resulting in a suboptimal equilibrium cooperation strategy. In this work, we show that reformulating an agent's policy to be conditional on the policies of its neighboring teammates inherently maximizes Mutual Information (MI) lower-bound when optimizing under Policy Gradient (PG). Building on the idea of decision-making under bounded rationality and cognitive hierarchy theory, we show that our modified PG approach not only maximizes local agent rewards but also implicitly reasons about MI between agents without the need for any explicit ad-hoc regularization terms. Our approach, InfoPG, outperforms baselines in learning emergent collaborative behaviors and sets the state-of-the-art in decentralized cooperative MARL tasks. Our experiments validate the utility of InfoPG by achieving higher sample efficiency and significantly larger cumulative reward in several complex, discrete- and continuous-space cooperative multi-agent domains.

7.1.1 Introduction and Motivation

Information sharing is key in building team cognition, and enables agents to cooperate and successfully achieve shared goals [241]. In addition to communication, individuals in high-performing human teams also benefit from the theory of mind [88] and making strategic decisions by recursively reasoning about the actions (strategies) of other human members [89]. Such hierarchical rationalization alongside with communication facilitate meaningful cooperation in human teams [90]. Similarly, collaborative MARL relies on meaningful cooperation among interacting agents in a common environment [91]. Most of the prior works on collaborative MARL are based on the maximum utility theory paradigm which assumes perfectly informed, rational agents [92]. Nevertheless, even under careful handcrafted or machine learned coordination policies, it is unrealistic and perhaps too strong to assume agents are perfectly rational in their decision-making [93, 94, 95, 96].

Recently, strong empirical evidence has shown that *Mutual Information* (MI) is a statistic that correlates with the degree of collaboration between pairs of agents [97]. Researchers have shown that information redundancy is minimized among agents by maximizing the joint entropy of agents' decisions, which in turn, improves the overall performance in MARL [267]. Therefore, recent work in MARL has sought to integrate entropy regularization terms as means of maximizing MI among interacting agents [98, 268, 99]. The formulaic calculation of MI relies upon the estimation of action-conditional distributions. In most prior work, agents are equipped with conventional state-conditional policies, and researchers employ techniques, such as variational inference, for estimating an action-conditional policy distribution to quantify MI [100, 98]. However, agents are not explicitly given the ability to reason about their teammates' action-decisions and, instead, have to learn implicitly from sparse rewards or hand-engineered regularization and auxiliary loss terms.

Contributions

In this work, we propose a novel information-theoretic, fully-distributed cooperative MARL framework, called InfoPG, by reformulating an agent's policy to be directly conditional on the policies of its instantaneous neighbors during Policy Gradient (PG) optimization. We study cooperative MARL under the assumption of bounded rational agents and leverage action-conditional policies into PG objective function to accommodate our assumption. By leveraging the *k*-level reasoning [269] paradigm from cognitive hierarchy theory, we propose a cooperative MARL framework in which naive, nonstrategic agents are improved to sophisticated agents that iteratively reason about the rationality of their teammates for decision-making. InfoPG implicitly increases MI among agents' *k*-level action-conditional policies to promote cooperativity. To learn collaborative behavior, we build InfoPG on a communicative fully-decentralized structure where agents learn to achieve consensus in their actions and maximize their shared utility by communicating with their physical neighbors over a potentially time-varying communication graph. We show the effectiveness of InfoPG across multiple, complex cooperative environments by empirically assessing its performance against several baselines. The primary contributions of our work are as follows:

1. We derive InfoPG, an information-theoretic policy gradient framework that leverages cognitive hierarchy and action-conditional policies for maximizing MI among agents and maximizing agents' individual rewards. We derive an analytical lower- and an upper-bound for MI estimated during InfoPG and provide mathematical reasoning underlying InfoPG's performance.

2. We propose a *fully*-decentralized graph-based communication and *k*-level reasoning MARL architecture based on LSTM networks to enable theory of mind for coordinating agents and maximizing their shared utility.

3. We propose a generalized variant of InfoPG and derive an MI upper-bound to modulate MI among agents depending on cooperativity of agents and environment feedback. We demonstrate the utility of this generalization in solving an instance of the Byzantine Generals Problem (BGP), in a fully decentralized setting.

4. We present quantitative results that show InfoPG sets the SOTA performance in learning emergent cooperative behaviors by converging faster and accumulating higher team rewards than information-based and decentralized MARL benchmarks.

7.1.2 Related Work

Cooperative MARL studies can be subdivided into two main lines of research, (1) learning direct communication among agents to promote coordination [66, 67, 68, 69] and, (2) learning to coordinate without direct communication [70, 71, 10]. Our work can be categorized under the former. Hierarchical approaches are also prevalent for learning coordination in MARL [5, 73, 74, 9]. We consider MARL problems in which the task in hand is of cooperative nature and agents can directly communicate, when possible. Unlike these studies, however, we improve our interacting agents from coexisting to strategic by enabling the recursive *k*-level reasoning for decision-making.

Researchers have shown that maximizing MI among agents leads to maximizing the joint entropy of agents' decisions, which in turn, improves the overall performance in MARL [267, 98]. As such, prior work has sought to increase MI by introducing auxiliary MI regularization terms to the objective function [98, 268, 99]. These prior works adopt a centralized paradigm, making them less relevant to our fully decentralized training and execution setting. Model of Other Agents (MOA) was proposed by [99] as a decentralized approach that seeks to locally push the MI lower-bound and promote collaboration among neighboring agents through predicting next-state actions of other agents. In all of the mentioned approaches, the amount of MI maximization objective that should be integrated into the overall policy objective is dictated through a β regularization parameter. In our work, however, we reformulate an agent's policy to be directly conditional on the policies of its neighbors and therefore, we seek to reason about MI among agents in our PG update without ad-hoc regularization or reward shaping.

Among prior work seeking to enable k-level reasoning for MARL, [100] presented Probabilistic Recursive Reasoning (PR2), an opponent modeling approach to decentralized MARL in which agents create a variational estimate of their opponents' level $k-1$ actions and optimize a joint Q-function to learn cooperative policies without direct communication. [93] later extended the PR2 algorithm for generalized recursive depth of reasoning. In InfoPG, we establish the inherent connection between k-level reasoning and MI, a link that has not been explored in prior work. Moreover, we bypass the need for modeling other agents through direct communication and k-level action-conditional policies, and giving InfoPG agents the ability to recursively reason about their teammates' actions through received messages and with any arbitrary rationalization depths.

7.1.3 Preliminaries

Problem Formulation – We formulate our setup as a Multi-Agent Fully Decentralized POMDP (MAF-Dec-POMDP), represented by an 8-tuple $\langle \{\mathscr{G}_t\}_{t\geq 0}, \mathscr{N}, \mathscr{S}, \mathscr{A}, \Omega, \{\mathscr{R}^i\}_{i\in\mathscr{N}}, \mathscr{P}, \gamma \rangle$. $\mathscr{N}$ is the set of all interacting agents in the environment in which index i represents the index of an agent. $\mathscr{G}_t = \langle \mathscr{N}, \mathscr{E}_t \rangle$ is a time-varying, undirected communication graph in which agents $i, j \in \mathscr{N}$ are vertices and $\mathscr{E}_t \subseteq \{(i,j) : i,j \in \mathscr{N}, i \neq j\}$ is the edge set. The two agents i and j can only share information at time t if $(i,j) \in \mathscr{E}_t$. State space $\mathscr{S}$ is a discrete set of joint states, $\mathscr{A}$ represents the action space, Ω is the observation space, and $\gamma \in [0,1)$ is the temporal discount factor for each unit of time.

At each step, t, an agent, i, receives a partial observation, $o_t^i \in \Omega$, takes an action, $a_t^i \in \mathscr{A}$, and receives an immediate individual reward, $r_t^i \in \{\mathscr{R}^i\}_{i \in \mathscr{N}}$. Taking joint actions, $\bar{a}$, in the joint states, $\bar{s}$, leads to changing the joint states to $\bar{s}' \in \mathscr{S}$, according to the state transition probabilities, $\mathscr{P}\left(\bar{s}'|\bar{s},\bar{a}\right)$. Our model is *fully* decentralized since agents take actions locally and receive individual rewards for their actions according to their own reward function. Moreover, each agent is also equipped with a local optimizer to update its individual policy through its local reward feedback. Accordingly, we can reasonably assume that agents' choices of actions are conditionally independent given the current joint states [63]. In other words, if $\bar{\pi} : \mathscr{S} \times \mathscr{A} \rightarrow [0,1]$ is the joint state-conditional policy, we assume that $\bar{\pi}(\bar{a}|\bar{s}) = \Pi_{i \in \mathscr{N}} \pi^i(a^i|\bar{s})$. Note that Decentralized POMDPs are allowed to facilitate local inter-agent communication [63, 91, 270].

Policy Gradient (PG) and Actor-Critic (AC) Methods – The policy gradient methods target at modeling and optimizing the policy π^i directly by parametrizing the policy, $\pi_\theta^i(a_t^i|s_t)$. Actor-Critic (AC) is a policy gradient method in which the goal is to maximize the objective by applying gradient ascent and directly adjusting the parameters of policy, π_θ^i, through an *actor* network. The actor, updates the policy distribution in the direction suggested by a *critic*, which estimates the action-value function $Q^w(s_t, a_t^i)$ [260]. By the policy gradient [126], the gradient by which the objective in AC, $J(\theta)$, is maximized can be shown as $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta^i}\left[\nabla_\theta \log \pi_\theta^i(a_t^i|o_t^i) Q^w(\bar{s}_t, a_t^i)\right]$, where a_t^i and o_t^i are agent i's action and observation.

Mutual Information (MI) – MI is a measure of the reduction in entropy of a probability distribution, X, given another probability distribution Y, where $H(X)$ denotes the entropy of X and $H(X|Y)$ denotes the entropy of the conditional distribution of X given Y [271, 272]. By expanding the Shannon entropy of X and $X|Y$, we can compute the MI as in Equation 7.1.

$$I(X;Y) = H(X) - H(X|Y) = \sum_{y \in Y} p_Y(y) \sum_{x \in X} p_{X|Y=y}(x) \log\left(\frac{p_{X|Y=y}(x)}{p_X(x)}\right) \tag{7.1}$$

In our work, X and Y are distributions over actions given a specific state, for two interacting agents. In an arbitrary Markov game with two agents i and j and with policies π_i and π_j, if π_i gains MI by viewing π_j, then agent i will make a more informed decision about its sampled action and vice versa.

7.1.4 Mutual Information Maximizing Policy Gradient

In this section, we first present an algorithmic overview of the InfoPG framework and then introduce the InfoPG objective by covering the logistics of building an iterated k-level decision making strategy for agents with bounded rationality. We then explore the relation of InfoPG with MI and derive an analytical lower-bound on the MI between the action-conditional policies of interacting agents.

Algorithmic Overview

Consider an MAF-Dec-POMDP introduced in subsection 7.1.3, with N agents where each agent is equipped with an *encoding* and a *communicative* policy (see Figure 7.1). At the beginning of a new rollout, each agent receives a state observation from the environment and produces an initial action (i.e., a guess action) using its encoding policy. Each agent i has a neighborhood of agents it can communicate with, shown with $j \in \Delta_t^i$ where $|\Delta_t^i|$ is the number of agent i's physical neighbors (i.e., within close proximity). Next, depending on the level of k in the decision hierarchy, agents communicate their action guesses (high-dimensional latent distributions) with their neighbors k times and update their action guess iteratively using their communicative policy. The level-k action is then executed by all agents and a local reward is given to each agent separately. This process continues until either the environment is solved successfully, or some maximum number of cycles has been attained. For each timestep, t, of the policy rollout and for each agent i, the gradient of the log probability is computed, scaled by the instantaneous advantage, A_t^i, and the encoding and communicative policies are updated. This process repeats until convergence of the

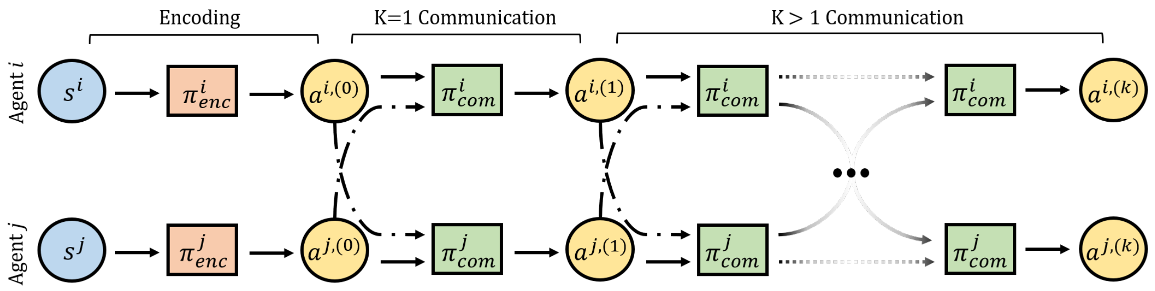


Figure 7.1: An instance of the information flow in a k-level decision hierarchy between two agents i and j for calculating their level-k strategies. Level-zero actions (e.g., $a^{i,(0)}$) represent the naive non-strategic actions.

cumulative discounted rewards across all agents. Please refer to Appendix, section D.1 for pseudocode and details of our training and execution procedures.

Deep Reasoning: Decision-Making Under k-Level Rationalization

We leverage from cognitive hierarchy theory [11] and strategic game theory [273], wherein each agent has k-levels of conceptualization of the actions its neighbors might take under bounded rationality. k-level reasoning assumes that agents in strategic games are not fully rational and therefore, through hierarchies of iterated rationalizability, each agent bases its decisions on its predictions about the likely actions of other agents [269]. According to k-level theory, strategic agents can be categorized by the *depth* of their strategic thought [274]. For example, consider the two agent case shown in Figure 7.1, with $k = 2$. At $k = 1$ agent i makes decisions based off its own observed state, and a guess of what agent j will do (Agent i will assume agent j is naive and non-strategic). A more sophisticated agent i with a strategy of level $k = 2$ however, makes decisions based off the rationalization that agent j has a level-one guess of i's strategy. Inductively, this line of reasoning can be extended to any k. Notice that the conditioning of agent i's policy on agent j's perceived actions is an action-conditional distribution. In InfoPG, we give agents the ability to communicate with their latent guess-action distributions in k iterated reasoning steps and rationalize their action decisions at level k to best respond to their teammates' level $k-1$ actions.

In the following, we represent the level of rationality for an agent by the superscript (k) where $k \in \mathbb{N}$. Denoting $\pi^{j,(k)}$ as the level-k policy of agent j, it can be shown that under the k-level reasoning, agent i's policy at level $k+1$, $\pi^{i,(k+1)}$, is precisely the best response of agent i to agent j's policy $\pi^{j,(k)}$ [92]. In other words, $\pi^{i,(k+1)} \in \text{BestResponse}(\pi^{j,(k)})$. In theory, this process can iteratively proceed until we obtain $\pi^{i,(k+2)} = \pi^{i,(k)}$, which corresponds to reaching the equilibrium strategies. In practice, however, for scenarios with many collaborating agents, the level of k is usually set to a reasonably small number for computational efficiency; since, in order to calculate the policy $\pi^{i,(k)}$, agent i must calculate not only its own policy at level k, but also all policies of all its neighbors for all $k \in \{1,2,3,\cdots,k-1\}$ at each time-step t.

InfoPG Objective

Our approach, InfoPG, equips each agent with an action-conditional policy that performs actions based on an iterated k-level rationalization of its immediate neighbors' actions. This process can be graphically described as presented in Figure 7.1, in which the information flow in a k-level reasoning between two agents i and j is shown. Note that in practice, any number of agents can be applied. Considering agent i as the current agent, we represent i's decision-making policy as $\pi^i_{tot} = [\pi^i_{enc}, \pi^i_{com}]$, in which $\pi^i_{enc}(a^i_t|o^i_t)$ is the state-conditional policy that maps i's observed states to actions. $\pi^i_{com}(a^{i,(k)}_t|a^{i,(k-1)}_t, a^{j,(k-1)})$ is the action-conditional policy that maps agent i's action at level $(k-1)$ along with the actions of i's neighbors (in this case agent j) at level $k-1$, to an action for agent i in the (k)-th level of decision hierarchy, $a^{i,(k)}_t$. Therefore, pursuant to the general PG objective, we define the basic form of our modified information-based objective, as in Equation 7.2, where Δ^i_t is the set of i's immediate neighbors in communication graph at time t and G_t is the return.

$$\nabla^{\text{InfoPG}}_\theta J(\theta) = \mathbb{E}_{\pi^i_{tot}}\left[G^i_t(o^i_t, a^i_t) \sum_{j\in\Delta^i_t} \nabla_\theta \log(\pi^i_{tot}(a^{i,(k)}_t | a^{i,(k-1)}_t, a^{j,(k-1)}_t, \ldots, a^{i,(0)}_t, a^{j,(0)}_t, o^i_t)) \right] \tag{7.2}$$

Equation 7.2 describes the form of InfoPG's objective function. Depending on the use case, we can replace the return, G_t^i, with action-values, Q_t^i, shown in Equation 7.3, and present the InfoPG as a Monte-Carlo PG method. We can also replace the returns with the advantage function, A_t^i, as shown in Equation 7.4, and present the AC variant of the InfoPG objective [126].

$$G_t^i(o_t^i, a_t^i) = Q_t^i(o_t^i, a_t^i) \quad \text{s.t.} \quad Q_t^i(o_t^i, a_t^i) \geq 0 \tag{7.3}$$

$$G_t^i(o_t^i, a_t^i) = A_t^i(o_t^i, a_t^i) = Q_t^i(o_t^i, a_t^i) - V_t^i(o_t) \tag{7.4}$$

Leveraging Equation 7.3 and Equation 7.4, we present two variants of the InfoPG objective. The first variant is the MI maximizing PG objective, which utilizes Equation 7.3. The non-negative action-value condition in Equation 7.3 implies non-negative rewards from the environment, a common reward paradigm utilized in prior work [275, 76, 276]. By applying this condition, InfoPG only moves in the direction of maximizing the MI between cooperating agents (see Theorem 6). We refer to the second variant of our InfoPG objective shown in Equation 7.4 as Advantage InfoPG (Adv. InfoPG), in which we relax the non-negative rewards condition. Adv. InfoPG modulates the MI among agents depending on cooperativity of agents and environment feedback (see Theorem 7).

Bayesian Expansion of the Policy

The action-conditional policy conditions an agent's action at the k-th level of the decision hierarchy on the actions of other agents at level $k-1$; however, to relate our k-level formulation to MI, we seek to represent an agent's action at a particular level k to be dependent on the actions of other agents at same level k. We present Theorem 5 to introduce the gradient term in the InfoPG objective in Equation 7.2 which relates level-k actions of the cooperating agents in their respective decision hierarchies. Please refer to the Appendix D, section D.5 for a detailed proof of Theorem 5.

Theorem 5 *The gradient of the log probability's level-k action-distribution in the InfoPG objective (Equation 7.2) for an agent, i, with neighbors $j \in \Delta_t^i$ and policy π_{tot}^i that takes the Maximum a Posteriori (MAP) action $a_t^{i,(k)}$, can be calculated iteratively for each level k of rationalization via Equation 7.5.*

$$\nabla \log(\pi_{tot}^i(a_t^{i,(k)} = MAP \mid a_t^{i,(k-1)}, a_t^{j,(k-1)}, \ldots, o_t^i)) \propto \nabla \log(\pi_{com}^i(a_t^{i,(k)} = MAP \mid a_t^{j,(k)} = MAP)) \tag{7.5}$$

InfoPG and Mutual Information Lower-Bound

Using the fact that $\nabla \log(\pi_{tot}^i(a_t^{i,(k)}|.))$ is directly proportional to $\nabla \log(\pi_{com}^i(a_t^{i,(k)}|a_t^{j,(k)}))$ from Eq. Equation 7.5, we can show that the gradient of our communicative policy implicitly changes MI. Since MI is empirically hard to estimate, we instead derive a lower-bound for MI which is dependent on the action-conditional policy of an agent. We show that increasing the probability of taking an action from the action-conditional policy will increase the derived lower-bound, and consequently, the MI.

Theorem 6 *Assuming the actions $(a_t^{i,(k)})$ and $a_t^{j,(k)}$ to be the Maximum a Posteriori (MAP) actions, the lower-bound to MI, $I^{(k)}(i;j)$, between any pair of agents i and j that exist in the communication graph $\mathscr{G}_t$ can be calculated w.r.t to agent i's communicative policy as shown in Equation 7.6.*

$$\pi_{com}^i(a_t^{i,(k)}|a_t^{j,(k)}) \log(\pi_{com}^i(a_t^{i,(k)}|a_t^{j,(k)})) \leq I^{(k)}(i;j) \tag{7.6}$$

Proof – Without loss of generality, we consider two agents $i, j \in \mathscr{N}$ with action-conditional policies $\pi^i(a^i|a^j)$ and $\pi^j(a^j|a^i)$ (note that the time, t, and the rationalization level, (k), indices are removed for notational brevity). We refer to the marginalizations of i's and j's action-conditional policies as *priors* which can be denoted as $p(a^i) = \sum_{a^j \in \mathscr{A}} \pi^i(a^i|a^j)$ and $p(a^j) = \sum_{a^i \in \mathscr{A}} \pi^j(a^j|a^i)$. We assume uniformity of the priors, as done previously by [277], such that $p(a^i) = p(a^j) = \frac{1}{|\mathscr{A}|}$, where $|\mathscr{A}|$ is the action-space dimension. For a detailed discussion on the validity of the uniformity of priors assumption, please refer to the Appendix, section D.6. Since MI is a marginalization across all actions in the action-space, a lower-bound exists at a particular a^i_{max}, which is the MAP action. As such, starting from the basic definition of MI in Equation 7.1, we derive:

$$\begin{aligned} I(i;j) &= \sum_{a^j \in \mathscr{A}} p(a^j) \sum_{a^i \in \mathscr{A}} \pi^i(a^i|a^j) \log\left(\frac{\pi^i(a^i|a^j)}{p(a^i)}\right) && (7.7) \\ &= \sum_{a^j \in \mathscr{A}} \frac{1}{|\mathscr{A}|} \sum_{a^i \in \mathscr{A}} \pi^i(a^i|a^j) \log(|\mathscr{A}| \pi^i(a^i|a^j)) && (7.8) \\ &\geq \frac{1}{|A|}[\sum_{a_j}\sum_{a_i} \pi^i(a^i|a^j) log(\pi^i(a^i|a^j))] && (7.9) \\ &\geq \pi^i(a^i_{\max}|a^j) log(\pi^i(a^i_{\max}|a^j)) && (7.10) \end{aligned}$$

We now seek to relate the last term in Equation 7.10 to the gradient term $\nabla \log(\pi^i_{com}(a^{i,(k)}_t|a^{j,(k)}_t))$ and variation of MI. By monotonicity of log, maximizing $\log(\pi^i(a^i_{max}|a^j))$ is equivalent to maximizing the $\pi^i(a^i_{max}|a^j)$ term. Therefore, according to Theorem 6 and Equation 7.10, the gradient updates will raise our lower-bound, which will maximize the MI. Since the sign of environment rewards and therefore, the sign of $Q_t(o_t, a_t)$ in Equation 7.3 is strictly non-negative, the gradient ascent updates will always move $\log(\pi^i_{com}(a^{i,(k)}_t|a^{j,(k)}_t)$ either up or not-at-all, which will have a proportional effect on the MI given the lower-bound. Note that we leveraged the non-negative reward condition so that the rationalization of $a^{i,(k)}_t$ given $a^{j,(k)}_t$ is only non-negatively reinforced. As such, if agent i's action-conditional policy on agent j's action does not obtain a positive reward from the environment, the lower-bound of MI stays constant, and thus so does MI. Conversely, if a positive reward is received by the agent, then the lower-bound of MI will strictly increase, leading to lowering the conditional entropy for taking the action that yielded the positive feedback from the environment.

Advantage InfoPG and the Byzantine Generals Problem

While continually maximizing MI among agents is desired for improving the degree of coordination, under some particular collaborative MARL scenarios such MI maximization may be detrimental. We specifically discuss such scenarios in the context of Byzantine Generals Problem (BGP). The BGP describes a decision-making scenario in which involved agents must achieve an optimal collaborative strategy, but where at least one agent is corrupt and disseminates false information or is otherwise unreliable [278]. BGP scenarios are highly applicable to cooperative MARL problems where there exists an untrainable fraudulent agent with a bad policy (e.g., random) in the team. Coordinating actions with such a fraudulent agent in a collaborative MARL setting can be detrimental. We note that BGPs are particularly challenging to solve in the fully decentralized settings [279, 280].

Here, we elaborate on our Adv. InfoPG variant introduced in Equation 7.4 and show its utility for intelligently modulating MI depending on the cooperativity of agents. Intuitively, the advantage function evaluates how much better it is to take a specific action compared to the average, general action at the given state. Without the non-negative reward condition in InfoPG, the Adv. InfoPG objective in Equation 7.4 does not *always* maximize the MI locally, but, instead, benefits from both positive and negative experiences as measured by the advantage function to increase the MI in the long run. Although instantaneous experiences may result in a negative advantage, A_t^i, and reducing the MI lower-bound in Equation 7.6, we show that Adv. InfoPG in fact *regularizes* an MI upper-bound, making it suitable for BGP scenarios while also having the benefit of learning from larger number of samples. In Adv. InfoPG we equip each agent with an action-conditional policy and show that under k-level reasoning, the tight bounds of MI between agents is regularized (shifted up and down) depending on the sign of the received advantage. We note that despite this local MI regularization, in the long run we expect the MI bounds to increase since policy gradient seeks to maximize local advantages during gradient ascent.

Upper-Bound of MI – Here, we derive an MI upper-bound dependent on the action-conditional policy of an agent and show that the gradient updates in Adv. InfoPG have a proportional effect on this upper-bound. We show that under k-level rationalization, the tight bound of MI between agents is regularized (shifted up or down) depending on the sign of the received advantage value, A_t^i.

Theorem 7 *Assuming the same preliminaries as in Theorem 6, the upper-bound to MI, $I^{(k)}(i;j)$, between agents i and j w.r.t agent i's level-k action-conditional policy can be calculated as in Equation 7.11.*

$$I^{(k)}(i;j) \leq 2\log(|\mathscr{A}|) + 2\log(\pi^i_{com}(a_t^{i,(k)}|a_t^{j,(k)})) \tag{7.11}$$

Proof – We start from the definition of conditional entropy. The conditional entropy is an expectation across all a^i and considering the fact that the $-\log(.)$ is a convex function, Jensen's inequality [281] can be applied to establish an upper-bound on conditional entropy. We derive:

$$H(\pi^i|\pi^j) = -\sum_{a^i\in\mathscr{A}} \pi^i(a^i|a^j)\log(\pi^i(a^i|a^j)) = \sum_{a^i\in\mathscr{A}} \pi^i(a^i|a^j)(-\log(\pi^i(a^i|a^j))) \tag{7.12}$$

$$\xrightarrow{\text{Jensen's inequality}} H(\pi^i|\pi^j) \geq -\log(\sum_{a^i\in\mathscr{A}} \pi^i(a^i|a^j)^2) \tag{7.13}$$

Now, we leverage the basic MI definition in Equation 7.1. We note that $H(p(a^i))$ has a constant value of $\log(|\mathscr{A}||)$ given the uniform prior assumption. Accordingly, plugging in the bound in Equation 7.13 and evaluating at the MAP action results in an upper-bound for MI, as shown below.

$$I(i;j) = H(p(a^i)) - H(\pi^i|\pi^j) = -H(\pi^i|\pi^j) + \log(|\mathscr{A}|) \leq \log(\sum_{a^i\in\mathscr{A}} \pi^i(a^i|a^j)^2) + \log(|\mathscr{A}|) \tag{7.14}$$

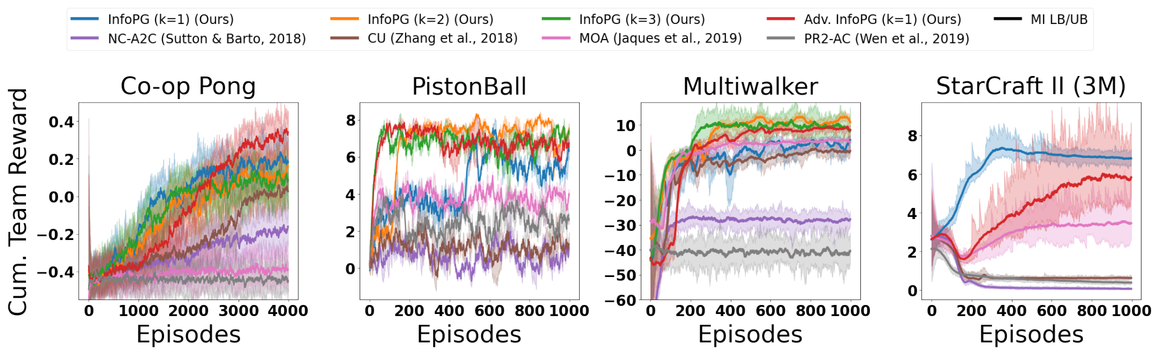
$$\leq \log\left(|\mathscr{A}|\pi^i(a^i_{max}|a^j)^2\right) + \log(|A|) \leq 2\log(|\mathscr{A}|) + 2\log\left(\pi^i(a^i_{max}|a^j)\right) \tag{7.15}$$

Considering the Adv. InfoPG objective in Equation 7.4, depending on the sign of A_t^i, the gradient ascent either increases or decreases $\log(\pi_{com}^i(a_t^{i,(k)}|a_t^{j,(k)}))$, which will have a proportional regulatory effect on the MI given our bounds in Equation 7.6 and Equation 7.11. Specifically, when agent i receives negative advantage from the environment, it means agent i's reasoning of $a_t^{i,(k)}$ given $a_t^{j,(k)}$, resulted in a negative outcome. Therefore, to reduce agent j's negative influence, our gradient updates in Equation 7.4 will decrease the MI upper-bound between i and j so that the conditional entropy at level k is increased. This bears similarity to Soft actor-critic [226], where regularization with entropy allows agents to *explore* more actions. As such, during Adv. InfoPG updates, the MI constraint becomes adaptive to instantaneous advantage feedback. Such property can be effective in BGP scenarios to reduce the negative effects of misleading information received from a fraudulent (or broken in other applications) agent.

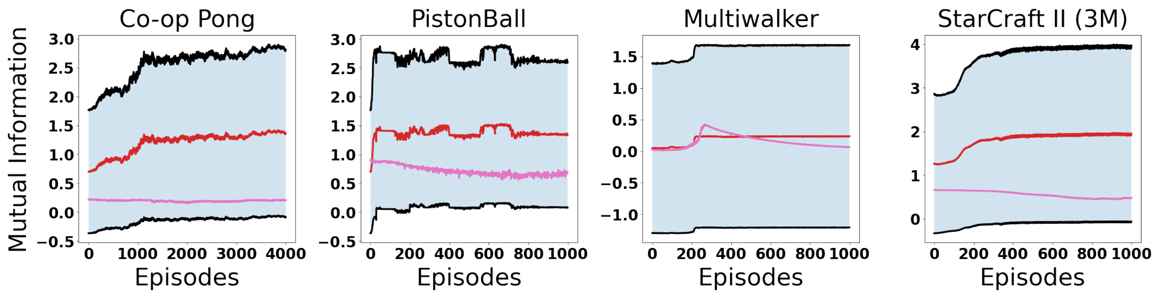
7.1.5 Empirical Evaluation

Evaluation Environments – We empirically validate the utility of InfoPG against several baselines in four cooperative MARL domains that require high degrees of coordination and learning collaborative behaviors. Our testing environments include: (1) Cooperative Pong (Co-op Pong) [282], (2) Pistonball [282], (3) Multiwalker [283, 282] and, (4) StarCraft II [284], i.e., the 3M (three marines vs. three marines) challenge. We modified the reward scheme in all four domains to be individualistic such that agents only receive a local reward feedback as per our MAF-Dec-POMDP formulation in subsection 7.1.3. For environment descriptions and details, please refer to Appendix D, section D.8.

Baselines – We benchmark our approach (both InfoPG in Equation 7.3 and Adv. InfoPG in Equation 7.4) against four *fully-decentralized* baselines: (1) Non-communicative A2C (NC-A2C) [126], (2) Consensus Update (CU) [63], (3) Model of Agents (MOA) [99] and, (4) Probabilistic Recursive Reasoning (PR2) [100]. In the NC-A2C, each agent is controlled via an individualized actor-critic network without communication. The CU approach shares



(a) Training Performance



(b) Mutual Information Variations

Figure 7.2: (Top Row) Team rewards obtained across episodes as training proceeds. The shaded regions represent standard error. Our Adv. InfoPG continually outperforms all baselines across all domains and in both training and testing (see Table 7.1). (Bottom Row) The MI ablation study results, comparing the MI variations between InfoPG and MOA (MI-based baseline) where InfoPG demonstrates a higher final average MI estimate across all domains. The shaded blue region represents the area between InfoPG's lower and upper bounds on MI.

the graph-based communication among agents with InfoPG but lacks the k-level reasoning in InfoPG's architecture. Thus, the CU baseline is communicative and non-rational. MOA, proposed by [99], is a decentralized cooperative MARL method in which agents benefit from action-conditional policies and an MI regularizer dependent on the KL-Divergence between an agent's prediction of its neighbors' actions and their true actions. PR2, proposed by [100] is an opponent modeling approach to decentralized MARL in which agents create a variational estimate of their opponents' level $k-1$ actions and optimize a joint Q-function to learn cooperative policies through k-level reasoning without direct communication.

7.1.6 Results, Ablation Studies, and Discussion

In this section, we assess the performance and efficiency of our frameworks in the four introduced domains and against several recent, fully-decentralized cooperative MARL approaches. Following an analysis of performance under *k*-level reasoning and an MI ablation study, we present a case-study, namely the *fraudulent agent experiment*, to investigate the utility of our MI-regularizing InfoPG variant (i.e. Adv. InfoPG in Equation 7.4) against MI-maximizing baselines, such as MOA [99] and InfoPG, in BGP scenarios. We provide further ablation studies, such as a level-*k* policy interpretation for InfoPG and a scalability analysis in the Appendix, section D.7.

Baseline Comparison – The top row in Figure 7.2 depicts the team rewards obtained by each method across episodes as training proceeds. In all four domains, both InfoPG and Adv. InfoPG demonstrate sample-efficiency by converging faster than the baselines and achieving higher cumulative rewards. Table 7.1 presents the mean (±standard error) cumulative team rewards and steps taken by agents to win the game by each method at convergence. Table 7.1 shows that InfoPG and Adv. InfoPG set the state-of-the-art for learning challenging emergent cooperative behaviors in both discrete and continuous domains. Note that, while in Pistonball fewer number of taken steps means better performance, in Co-op Pong and Multiwalker more steps shows a superior performance.

Mutual Information Variation Analysis – The bottom row in Figure 7.2 shows our MI study results comparing the MI variations between InfoPG and MOA (the MI-based baseline). The MI for InfoPG is estimated as the average between lower and upper bounds defined in Equation 7.6 and Equation 7.11. As depicted, InfoPG demonstrates a higher final average MI estimate across all domains. Note the concurrency of InfoPG's increase in MI estimates (bottom row) and agents' performance improvements (Figure 7.2, top row). This concurrency supports our claim that our proposed policy gradient, InfoPG, increases MI among agents which results in learning emergent collaborative policies and behavior.

Deep Reasoning for Decision Making: Evaluating *k*-Level InfoPG – We evaluate

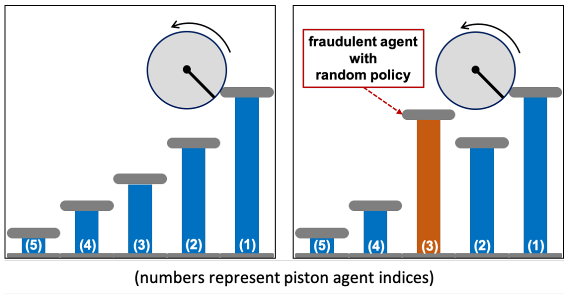
Table 7.1: Reported results are Mean (Standard Error) from 100 testing trials. For all tests, the final training policy at convergence is used for each method and for InfoPG and Adv. InfoPG, the best level of k is chosen.

Domain	**InfoPG**		**Adv. InfoPG**		MOA		CU		NC-A2C		PR2-AC	
	$\mathscr{R}$	#Steps	$\mathscr{R}$	#Steps	$\mathscr{R}$	#Steps	$\mathscr{R}$	#Steps	$\mathscr{R}$	#Steps	$\mathscr{R}$	#Steps
Co-op Pong	**0.127**	**202.9**	**0.25**	**212.9**	-0.3	58.2	0.13	152.0	0.04	102.925	-0.84	36.8
	(0.00)	**(1.35)**	**(0.00)**	**(1.24)**	(0.00)	(0.32)	(0.00)	(0.96)	(0.00)	(0.93)	(0.00)	(0.34)
Pistonball	**7.47**	**17.44**	**7.33**	**28.31**	4.10	91.66	1.06	136.3	0.89	138.3	1.90	140.4
	(0.02)	**(0.29)**	**(0.02)**	**(0.43)**	(0.03)	(0.76)	(0.04)	(0.83)	(0.04)	(0.83)	(0.02)	(0.68)
Multiwalker	**3.56**	**474.2**	**11.81**	**500.0**	0.66	489.3	-1.7	490.2	-66	80.75	-84	94.17
	(0.20)	**(1.39)**	**(0.11)**	**(0.80)**	(0.15)	(1.09)	(0.26)	(1.30)	(0.18)	(0.20)	(0.37)	(1.40)
StarCraft II	**4.40**	**30.79**	**3.73**	**44.5**	2.78	27.7	0.24	58.72	0.00	60.0	0.64	27.4
	(0.01)	**(0.05)**	**(0.02)**	**(0.13)**	(0.02)	(0.07)	(0.00)	(0.06)	(0.00)	(0.00)	(0.00)	(0.08)

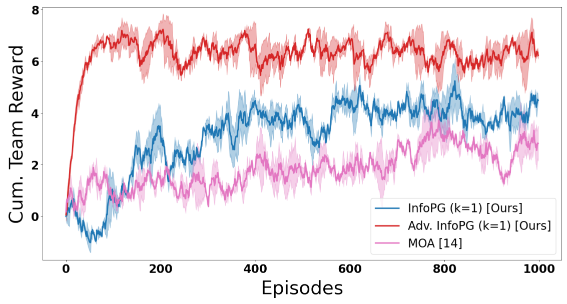
the performance of our method for deeper levels of reasoning $k \in \{2,3\}$ in Co-op Pong, Pistonball and Multiwalker domains. The training results are presented in Figure 7.2. As discussed in subsubsection 7.1.4, in the smaller domain with fewer collaborating agents, the Co-op Pong, agents reach the equilibrium cooperation strategy (i.e., $\pi^{i,(k+2)} = \pi^{i,(k)}$) even with one step of reasoning, and increasing the level of k does not significantly change the performance. However, in the more complex domains with more agents, Pistonball and Multiwalker, as the level of rationalization goes deeper in InfoPG (i.e., $k = 2$ and $k = 3$), agents can coordinate their actions better and improve the overall performance.

The Fraudulent Agent Experiment: Regularising MI – To assess our Adv. InfoPG's (Equation 7.4) regulatory effect based on agents' cooperativity, we perform an experiment, demonstrated in Figure 7.3a, which is performed in the Pistonball domain and is intended to simulate an instance of the BGP in which the middle piston is equipped with a fully random policy throughout the training (i.e., the Byzantine piston is not controllable by any InfoPG agent). Maximizing the MI with this fraudulent agent is clearly not desirable and doing such will deteriorate the learning performance.

Figure 7.3b presents the fraudulent agent experiment training results for Adv. InfoPG, InfoPG and MOA. As shown and comparing with Figure 7.2, existence of a fraudulent agent significantly deteriorates the learning performance in MOA and InfoPG as these approaches always seek to maximize the MI among agents. This is while InfoPG still outperforms MOA since by only using strictly non-negative rewards, the coordination in InfoPG is only positively reinforced, meaning that InfoPG only increases MI when the reward feedback is positive. Adv. InfoPG shows the most robust performance compared to InfoPG and MOA. Adv. InfoPG modulates the MI depending on the observed short-term coordination performance. As discussed in subsubsection 7.1.4, if the advantage is negative, the gradient ascent in Adv. InfoPG will decreases the MI upper-bound between agents, leading to increasing the conditional entropy and taking more exploratory (i.e., less coordinated) actions.


fraudulent agent with random policy
(5)
(4)
(3)
(2)
(1)
(numbers represent piston agent indices)

(a) The Fraudulent Agent Experiment: Scenario



(b) The Fraudulent Agent Experiment: Result

Figure 7.3: The fraudulent agent experiment scenario (Figure 7.3a) and training results (Figure 7.3b) in the Pistonball domain, comparing the team reward performance for Adv. InfoPG (Equation 7.4), InfoPG (Eq.Equation 7.3) and MOA.

7.1.7 Limitations and Future Work

Useful MI between agents becomes hard to capture in cases, such as Co-op Pong domain, where an agent's action influences its neighbors with some delay. Moreover, MI maximization by applying the strictly non-negative reward condition in InfoPG objective (Equation 7.3) comes at the cost of zeroing out negative experiences which may have an impact on sample-efficiency (also relevant to other MI-based methods such as MOA [99]). Moreover, for environments with many interacting agents, as the rationalization level, *k*, increases, the computational overhead of calculating the action-conditional distributions also raises. Lastly, the action-conditional policies in InfoPG's architecture are limited to homogeneous action-spaces.

7.1.8 Conclusion

We leverage iterated *k*-level reasoning from cognitive hierarchy theory and present a collaborative, fully-decentralized MARL framework which explicitly maximizes MI among cooperating agents by equipping each agent with an action-conditional policy and facilitating iterated inter-agent communication for hierarchical rationalizability of action-decisions. We analytically show that the design of our MI-based PG method, increases an MI lower-bound, which coincides with improved cooperativity among agents. We empirically show InfoPG's superior performance against various baselines in learning cooperative policies. Finally, we demonstrate that InfoPG's regulatory effect on MI makes it Byzantine-resilient and capable of solving BGPs in fully-decentralized settings.

CHAPTER 8
LEARNING HETEROGENEOUS TEAMING FROM HUMAN DEMONSTRATION

Traditional controller design for deploying robot teams typically consists of an army of expert consultants who design, build, and program robots for each application [285]. As such, in the previous chapter, we investigated data-based and RL to automatically learn such distributed controllers for multi-robot team coordination. Nevertheless, MARL also suffers from several problems, including designing an expressive reward function [286] and are typically hard to train with increased domain complexity [285].

As multi-robot systems become increasingly prevalent in our communities and workplace, aligning the values motivating their behavior with human values is critical [287]. Learning from Demonstration (LfD) attempts to learn the correct behavior (policy) from a set of expert-generated demonstrations rather than a reward function, which can result in lower sample complexity and directly learning human's preferred strategy [288, 287]. As such, in this chapter, we propose a novel Multi-agent LfD approach to learning high-quality collaborative multi-robot policies directly from human-expert generated data [289].

8.1 Mixed-Initiative Multi-Agent Apprenticeship Learning for Human Training of Multi-Robot Teams

Extending recent advances in Learning from Demonstration (LfD) frameworks to multi-robot settings poses critical challenges such as environment non-stationarity due to partial observability which is detrimental to the applicability of existing methods. Although prior work has shown that enabling communication among agents of a robot team can alleviate such issues, creating inter-agent communication under existing Multi-Agent LfD (MA-LfD) frameworks requires the human expert to provide demonstrations for both environment actions and communication actions, which postulates an existing efficient communication strategy on a known message spaces. To address this problem, we propose Mixed-Initiative Multi-Agent Apprenticeship Learning (MixTURE). MixTURE enables robot teams to learn from a human expert-generated data a preferred strategy to accomplish a collaborative task, while simultaneously learning end-to-end emergent inter-agent communication to enhance team coordination. The key ingredient to MixTURE's success is automatically learning an inter-agent communication policy, enhanced by a mutual-information maximizing reverse model, that rationalizes the underlying expert demonstrations without the need for human generated data or an auxiliary reward function. MixTURE outperforms a variety of baselines on data generated by human experts as well as expert-designed heuristics by achieving 22% higher performance and more than $10\times$ lower sample complexity in complex heterogeneous domains. MixTURE is the first to enable learning high-quality multi-robot collaborative policies directly from real human generated data, resulting in 44% less human workload, and 46% higher usability score.

8.1.1 Introduction

In recent years, Multi-Agent Reinforcement Learning (MARL) has been predominantly used by researchers to optimize a reward signal and learning multi-robot tasks. Nevertheless, RL generally suffers from key limitations such as the difficulty of designing an expressive and suitable reward function for complex tasks [110, 286] which can lead to undesirable robot behavior [290, 110, 113], high sample complexity [291], and safety concerns due to direct robot-environment interactions for optimizing the policy [113]. These problems are further exacerbated in multi-robot scenarios where inter-robot interactions and environment dynamics can be more complex and task descriptions and objectives more ambiguous [110]. As such, accurate models of human strategies and behaviors achieved via imitation and inverse RL methods are increasingly important for safely and effectively deploying autonomous systems and align values motivating their behaviors with human values [287, 292].

As multi-robot systems become increasingly prevalent in our communities and workplace, aligning the values motivating their behavior with human values is critical [287]. Learning from Demonstration (LfD) attempts to learn the correct behavior (policy) from a set of expert-generated demonstrations rather than a reward function, which can result in lower sample complexity and directly learning human's preferred strategy [288, 287]. Different variants of LfD include Imitation Learning (IL) and Inverse Reinforcement Learning (IRL). IL intends to learn a direct state-action mapping from a set of demonstrations, either offline through Behavioral Cloning (BC) [106] or online from human critique (i.e., DAgger [107]). IRL [108], on the other hand, assumes that the expert is approximately optimizing an underlying reward function and aims to infer a reward that rationalizes the demonstrations [110]. Unfortunately, extending these single-agent paradigms to multi-robot settings poses several challenges such as environment non-stationarity and existence of multiple equilibrium solutions (an agent's optimal policy depends on other agents' policies) [293, 294, 110]. One can adopt such frameworks directly in a centralized multi-agent system. However, centralized systems are not scalable, are prone to single-node failure, and pose significant computation overhead, and therefore, decentralized approaches (e.g., limited-range communication and local computations) have been more desired in multi-robot systems [295].

Prior work has shown that enabling communication among agents of a robot team creates a shared mental model of joint action-spaces and task objectives and therefore, allowing coordinated action decisions [239]. Building a truly collaborative robot team capable of accomplishing more than just the sum of its components' abilities requires developing a social dexterity and understanding among the robots of a team, where each agent needs to reason about its counterparts' intentions, beliefs, and goals to take appropriate actions at the right time in response to their teammates' actions. Such social dexterity and coordination can be achieved through communication without which, a multi-robot team can easily fail in tasks due to challenges such as partial observability and environment dynamicity [296, 13]. While MARL has been shown to be capable of learning such communication policies through environment interactions [68, 66, 13], it suffers from high sample complexity and reward shaping. On the other hand, although LfD can resolve these problems in RL, the task of MA-LfD can be even more onorous for the humans as they have to control multiple robots and imagine and simulate an appropriate theory of mind to create a communication strategy for the robot team.

Nevertheless, introducing the inter-agent communication under current state-of-the-art (SOTA) Multi-Agent LfD (MA-LfD) frameworks such as Multi-Agent Generative Adversarial IL (MA-GAIL) [110] or Multi-Agent Adversarial IRL (MA-AIRL) [111] requires the human expert to provide demonstrations for both *environment actions* and *communication actions*. Such approaches assume that the human expert has access to an efficient communication strategy on a known finite message space in addition to a known strategy for taking environment actions. This assumption, however, is unrealistic since in a general dynamic environment with multiple interacting agents there can be many unforeseen states and designing a cohesive and comprehensive inter-agent communication protocol seems exhaustive. Even if such efficient communication strategy exists, demonstrating both the task strategy as well as the communication strategy could potentially pose significant workload on the human expert, which in turn can affect the performance and quality of demonstrations provided by the expert. These issues become even more severe when the robot team is heterogeneous or of composite nature (i.e., agents with different observation- and action-spaces as well as different tasks and objectives) where agents must rely on communication to operate and fulfill their tasks correctly [297, 13, 87].

To address these challenges, we develop a distributed MA-LfD framework to efficiently incorporate a human expert's domain-knowledge of teaming strategies for collaborative robot teams and directly learn team coordination policies from expert human teachers. To this end, we propose Mixed-Initiative Multi-Agent Apprenticeship Learning (MixTURE). MixTURE enables robot teams to learn an expert's preferred strategy to act in an environment, while simultaneously learning end-to-end emergent communication for the robot team to enhance team coordination, without the need for human generated data or an auxiliary reward function. To improve the quality of the learned inter-agent communication protocol, we reduce the entropy of a generated message given joint states and actions by maximizing the Mutual Information (MI) between messages and joint states. We demonstrate through empirical evaluation and a human subject experiment that MixTURE benefits from the merits of LfD methods over RL such as reward function independence and low sample complexity, while significantly alleviating the human demonstrators' workload and time required to provide demonstrations, as well as increasing the System Usability Score (SUS) and overall collaboration performance of the robot team. **Our key contributions are as follow**:

1. We propose the MixTURE framework for learning robot teaming strategies from human expert demonstrations while simultaneously learning inter-agent communication through online interactions during training, without the need for expert data or an auxiliary reward.

2. We develop an MIM-based emergent communication learning model which reduces the entropy of a generated message for an agent given joint state-observations.

3. We train and evaluate MixTURE on real human data collected in an IRB-approved human subject experiment with 55 subjects in a complex, heterogeneous firefighting domain. Our results certify MixTURE's ability to learn high-quality multi-agent coordination policies, by achieving 22% higher performance and more than $10\times$ lower sample complexity, despite the heterogeneity and variance within human dataset.

4. We investigate the effects of demonstrating both environment actions and communication actions on a human expert's workload, demonstration quality, and system usability score. Our results show that a high-workload demonstration process in classic MA-LfD approaches significantly ($p < .001$) reduces an expert's demonstration quality (measured by performance) and the system's usability score. MixTURE

significantly improves these results; increasing a human's performance and experience engaging in MA-LfD.

8.1.2 Related Work

Learning from Demonstration (LfD) explores techniques for learning a task policy from examples provided by a human teacher [101, 102]. Ho et al. [103] and Fu et al. [104] formulated the LfD problem under generative adversarial learning setting [105] to tackle the limitations in classic LfD frameworks such as BC [106], DAgger [107], and IRL [108]. Generative Adversarial IL (GAIL) [103] collects state-action pairs from executing the learned policy to shift the trajectories closer to the desired behavior. In GAIL, a *discriminator* model is trained to distinguish between state-action pairs provided by the expert and a deceiving *generator* model (i.e., the learned policy) that learns to imitate the expert. Standard RL algorithms are leveraged to optimize over the output of the discriminator (i.e., treated as a reward signal), encouraging the agent to match the expert-data in expectation, over full trajectories. Adversarial IRL (AIRL) [104] follows a setup similar to the GAIL but addresses the reward signal ambiguity in GAIL by leveraging a specific discriminator structure.

The literature for Multi-Agent LfD (MA-LfD) primarily aims to address the complexity of simultaneously training multiple agents. In [109], a coordinated IL approach is proposed which learns a latent coordination model along with the individual policies. In [110] the single-agent GAIL framework, described above, is extended for multi-agent scenarios along with a practical actor-critic method for multi-agent imitation. Yu et al. [111] extend the AIRL method to the multi-agent settings and propose a scalable framework. In [112] a scalable multi-agent LfD approach is proposed where a model-based heuristic method for automated swarm reorganization is leveraged to improve multi-agent task allocation problem. In [113] authors create an advising system to incorporate sub-optimal model-based heuristic policies to help improve MARL performance. More recently, Hoque et al. [114] proposed Fleet-DAgger , formalizing interactive fleet learning setting, in which multiple robots interactively query and learn from multiple human supervisors.

Nevertheless, applicability of these prior work in the collaborative multi-agent problems are considerably limited since none of these works explicitly address the inter-agent communication in complex domains where agents not only need to take task-related actions, but also need to communication and share information for coordination. Enabling inter-agent communication in these prior work requires the human expert to provide demonstrations for both environment actions and communication actions, which postulates an existing efficient communication strategy on a known message spaces. Additionally, none of these prior work consider a partially observable domain (common for realistic multi-robot systems) which necessitates the need for inter-agent communication and do not leverage real human-generated data for training to evaluate the approach against heterogeneity and variance in human data. These limitations can alleviate applicability of the mentioned works to multi-robot scenarios.

In our work, we address the limitations in prior work by relaxing the need for demonstrating a communication strategy by the expert. Using our method, a human expert can only teach the robot team how to accomplish a task collaboratively via demonstrations and the team will automatically learn a communication strategy suitable for the cooperation policy underlying the expert's demonstrations. The learned communication protocol will then help the robot team to deal with the partial observability, reasoning about action-decisions to best respond to teammates' policies, and alleviate the effects of environment non-stationarity. We also collect real human data and evaluate MixTURE's ability to cope with demonstration heterogeneity due to different expert styles and strategies.

8.1.3 Problem Formulation: General MA-LfD with Heterogeneous Agents

We ground our problem formulation in a Markov Game (MG) [298] generalized to include partial observability and heterogeneous agents. We define a set of heterogeneous agents in a *composite* robot team (i.e., composed of different classes of robots) as agents that can have arbitrarily different state-, observation-, and action-spaces which can also have different task objectives while their tasks are co-dependent on accomplishing an overarching mission. Accordingly, we define our generic MG as a 9-tuple: $\langle \mathscr{C}, \mathscr{N}, \{\mathscr{S}^{(c)}\}_{c\in\mathscr{C}}, \{\mathscr{A}^{(c)}\}_{c\in\mathscr{C}}, \{\Omega^{(c)}\}_{c\in\mathscr{C}}, \{\mathscr{O}^{(c)}\}_{c\in\mathscr{C}}, r, \mathscr{T}, \gamma\rangle$. $\mathscr{C}$ is set of all available agent classes in the composite robot team and the index $c \in \mathscr{C}$ shows the agent class. $\mathscr{N} = \sum_{\langle c\in\mathscr{C}\rangle} N^{(c)}$ is the total number of collaborating agents where $N^{(c)}$ represents the number of agents in class c. $\{\mathscr{S}^{(c)}\}_{c\in\mathscr{C}}$ and $\{\mathscr{A}^{(c)}\}_{c\in\mathscr{C}}$ are discrete joint sets of state- and action-spaces, respectively. $\{\Omega^{(c)}\}_{c\in\mathscr{C}}$ is the joint set of observation-spaces, including class-specific observations. Agents of the same *class* have similar $\mathscr{S}$, $\mathscr{A}$, and Ω. $\gamma \in [0,1)$ is the temporal discount factor for each unit of time and $\mathscr{T}$ is the state transition probability density function. At each timestep, t, each agent, j, of the c-th class can receive a partial observation $o_t^{c_j} \in \Omega^{(c)}$ according to some class-specific observation function $\{\mathscr{O}^{(c)}\}_{c\in\mathscr{C}}$: $o_t^{c_j} \sim \mathscr{O}^{(c)}(\cdot|\bar{s})$. Next, each agent, j, of class c, takes an action, $a_t^{c_j}$, forming a joint action vector $\bar{a} = \left(a_t^{1_1}, a_t^{1_2}, \cdots, a_t^{c_1}, \cdots, a_t^{c_j}\right)$. When agents take the joint action $\bar{a}$, in the joint state $\bar{s}$ and depending on the next joint-state, they receive an immediate reward, $r(\bar{s},\bar{a}) \in \mathbb{R}$, shared by all agents regardless of classes. Each agent, j, of a class, c, achieves its own objective by sampling actions from a stochastic policy $\pi_j^{(c)}$. The objective of each agent is then to maximize the team return (expected sum of discounted rewards), i.e., $\mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t\right]$.

To directly learn the human's preferred strategy (i.e., value alignment) and resolve the reward specification problems [286], in LfD, we leverage a demonstration dataset, D, provided by an expert, rather than the ground truth reward signal r. Note that, unlike [111, 110] or [299], we do not assume multiple human experts in our MG to avoid the need for further coordination amongst the experts, which cane be time consuming and expensive. D is a set of trajectories $\{\tau_j^c\}_{j=1}^{\mathscr{N}^{(c)}}$, where $\tau_j^c = \{(o_t^{c_j}, a_t^{c_j})\}_{t=1}^{T}$ is an expert trajectory collected by sampling $a_t^{c_j} \sim \pi_E(a_t^{c_j}|\bar{o}_t)$ in which π_E is the expert policy and $\bar{o}_t$ is the joint observation that the expert has access to at time t. We further assume that D contains the entire supervision to the learning algorithm (i.e., no online interactions during training). We build the MixTURE architecture on the generative adversarial training [105], leading to the GAIL framework.

Our distributed GAIL objective underlying MixTURE is shown in Equation 8.1 where $\mathscr{D}_\theta^{(c_j)}$ is a local discriminator that classifies expert and policy trajectories for agent j of a class c, and $\pi_\phi^{(c_j)}$ is the parameterized policy of agent j of a class c.

$$L_{\mathscr{D}_\theta^{(c)}} = -\mathbb{E}_{\tau\sim\pi_E,(\bar{o},\bar{a})\sim\tau}\left[\log \mathscr{D}_\theta^{(c_j)}(\bar{o},\bar{a})\right] - \mathbb{E}_{\tau\sim\pi_\phi^{(c_j)},(\bar{o},\bar{a})\sim\tau}\left[\log\left(1-\mathscr{D}_\theta^{(c_j)}(\bar{o},\bar{a})\right)\right] \quad (8.1)$$

According to [105], under the GAIL objective in Eq. Equation 8.1 and at optimality, the distribution of generated state-action pairs by $\bar{\pi}_\phi$ should match the distribution of demonstrated state-action pairs.

8.1.4 Mixed-Initiative Multi-Agent Apprenticeship Learning (MixTURE)

Motivation and Problem Overview

Consider a generic composite team of robots including agents with heterogeneous characteristics and task objectives. Without loss of generality, consider a robot team composed of *perception-only* and *action-only* robots. Under our problem formulation in Section subsection 8.1.3, perception robots and action robots create two separate *class* of agents that need to collaborate on an overarching. For instance, in an application of wildfire fighting, robots of the perception class (e.g., quadcopters) need to search an environment for firespots, while the action robots (e.g., fire-extinguishing ground robots) who cannot sense the environment are required to extinguish the firespots found by the perception robots [13, 10, 9, 53]. Note that neither of the robot classes are capable of accomplishing the task without the other class.

To teach a collaborative multi-agent coordination strategy to such a robot team, one can leverage demonstrations from a team of humans where each member is an expert in one of the class-specific tasks. Using a team of human experts, however, poses further challenges: (1) simultaneous access to several human experts is expensive and can be time consuming, and (2) a communication strategy (e.g., natural language) among the human demonstrators is also required for coordination, which can be challenging to translate to robot domain due to ambiguity, colloquialisms, and context-dependent use [300, 301]. Additionally, humans' communications might not make sense to the robot agents as humans could be unaware of all agents' full local state spaces.

Alternatively, we can leverage the demonstrations from a single human with full domain-knowledge regarding the entire mission objective. For example, a trainer/coach can play a simulated game of firefighting using the aforementioned perception and action robots and provide expert demonstrations for how to efficiently distributed and prioritize tasks for searching the environment and putting out the propagating firespots. The challenge of using a single human expert, however, is that in this case the human would also need to demonstrate communication-actions (i.e., what information should an agent broadcast at each state) on top of the environment-actions (i.e., moving around or dousing fire) to deal with environment dynamicity and agents' partial observability. Not only providing such communication-action demonstration leads to an increased action-space dimension and requires an expert heuristic communication strategy over a known message-space, humans would also have to maintain a theory-of-mind (ToM) of each agent, which increases workload greatly [302, 303].

To resolve this problem, we propose taking separate initiatives for teaching the robots in the team how to operate (environment actions, a_t, per observation, o_t) and how to communicate (communication message, z_t, per state, s_t) such that a human expert would only be required to provide environment-action demonstrations and the robot team would automatically unravel a suitable communication policy for the underlying expert demonstrations. We call our approach Mixed-Initiative Multi-Agent Apprenticeship Learning (MixTURE). Next, we describe the MixTURE architecture.

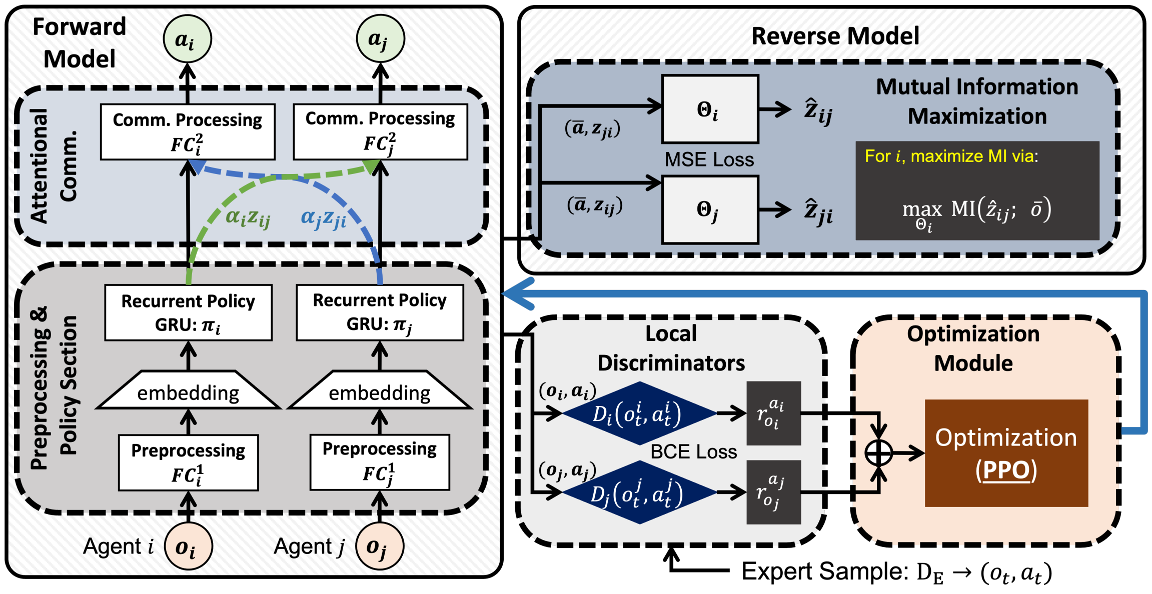


Figure 8.1: The proposed MixTURE architecture for human training of robot teams with MIM-based learned differentiable inter-agent communication for a two-agent example scenario. At each timestep, each agent generates an embedding from its local observation, which are then passed into local recurrent policies (to handle partial observability) for each agent. To learn from the expert demonstrations stored in the dataset D_E, we build a distributed multi-agent GAIL architecture (i.e., per agent discriminators, $\mathscr{D}_\theta$). We enable inter-agent communication by adding an attentional communication module enhanced by a MI maximization reverse model.

MixTURE Architecture

The proposed MixTURE architecture for human training of robot teams is shown in Fig. Figure 8.1 for a two-agent example scenario. At each timestep, each agent generates an embedding, representing agent's belief space, from its local observation. To handle agents' partial observability, the local observation embeddings are then passed into local recurrent policies (i.e., GRU) for each agent. Each GRU policy receives the preprocessed features from the local observations as well as its own hidden state from previous timesteps. Therefore, the policy output depends only on the history of local observations and actions.

To learn from the expert demonstrations stored in the dataset D_E, as shown in Fig. Figure 8.1, we build a distributed multi-agent GAIL architecture, similar to [110], where each agent is equipped with a parametrized discriminator, $\mathscr{D}_\theta$. The discriminators are trained via a Binary Cross Entropy (BCE) loss to distinguish between state-action pair samples from the expert dataset and those generated by the generator (i.e., local policies). The output of the discriminators are treated as local rewards, which are then combined to generate a shared reward signal for the team. Such a shared reward scheme incentifies collaboration and teaming behaviour among agents [69]. To learn the agent policies through this reward signal, we leverage the Proximal Policy Optimization (PPO) [304].

We enable inter-agent communication by adding an attentional communication module in which each agent is equipped with a fully-connected network that processes action-embeddings generated by the recurrent policies of an agent and those generated by its teammates (i.e., messages, z) to output an action-decision. We enable this message-passing by creating differentiable communication channels among agents. To maintain locality, these communication channels can be leveraged locally (i.e., a communication graph where edges only exist when robots are spatially within close proximity). For an agent, i, the action-embedding messages received from a teammate, j, are weighted by some learned attention coefficients, α_{ji}, to impose message importance. Therefore, the input message for agent i at time, t, can be computed as $m_t^{j\rightarrow i} = \sum_{j=1}^{\mathcal{N}_t(i)} \alpha_{ji} z_{ji}^t$ where $\mathcal{N}_t(i)$ represents the neighbors, j, of agent i at current timestep. in this equation, α_{ji} are the learned attention coefficients that are computed via Eq. Equation 8.2 where $\bar{W}_{att}$ are the learnable weights of the attention network, $\|$ represents concatenation, σ is an activation function nonlinearity, and $\bar{h}$ represents the hidden states. The Softmax function is used to normalize the coefficients across all neighbors j.

$$\alpha_{ji} = \text{softmax}_j(\sigma(\bar{W}_{att}[\omega\bar{h}_i \,\|\, \bar{m}^{j\rightarrow i}])) \tag{8.2}$$

Such attentional communication can enhance the action-decision quality, specially with increased number of agents or when the states may significantly vary in different parts of the environment [67]. Our entire communication module is differentiable, and messages are learned via backpropagation.

Mutual Information Maximization-Based Differentiable Communication

A challenge with the communication strategy learned via the end-to-end differentiable channels described in Section subsubsection 8.1.4 is that, the distribution of the messages sent by an agent, i, to a teammate, j, given the state-observations, $\rho(z_{ij}|\bar{o})$, can have a high variance. The desired behavior, however, would be to have a cohesive communication strategy where an agent is consistent with regards to sending a specific message when it observes relatively similar states at two different timesteps.

To resolve this issue, we propose maximizing the Mutual Information (MI) between an agent's outgoing message and the joint state-observations. The MI is a measure of the reduction in entropy of a probability distribution, X, given another probability distribution, Y, such that $I(X;Y) = H(X) - H(X|Y)$, where $H(X)$ denotes the entropy of X and $H(X|Y)$ is the conditional entropy of X given Y [271]. In our work, by maximizing the MI between the distribution of an agent's message, $\rho(z_{ij}|\bar{o})$, and the joint observations, we reduce the entropy over messages and encourage the model to be more consistent. Maximizing the MI in this way encourages z_{ij} to correlate with semantic features within the observation distribution (i.e., mode discovery) [305, 306].

Unfortunately, a direct MI Maximization (MIM) between the message distributions and joint observations as formulated above, $I(z_{ij};\bar{o}) = H(z_{ij}) - H(z_{ij}|\bar{o})$, is intractable as it requires access to the true posterior, $\rho(z_{ij}|\bar{o})$. Therefore, researchers employ the Evidence Lower Bound (ELBO) of the MI instead. As shown in prior work [305], by minimizing a Mean-Squared Error (MSE) loss between a sample from the current message embedding and the approximate posterior, modeled as a normal distribution with constant variance, is equivalent to maximizing the likelihood of the posterior. This is because with constant variance, the exponential function in approximate normal distribution is monotonic, and thus, minimizing the exponent will maximize the likelihood of the posterior [305].

In practice, we build a distributed reverse model in the MixTURE architecture (shown in Fig. Figure 8.1) to accommodate the mentioned MSE loss. The distributed reverse model for each agent has access to the local received messages as well as the global joint-actions taken by all agents such that the optimization results in a communication entropy-reduction mechanism at the team level. Since actions and generated messages are functions of the state-observations, each reverse model can take in the joint actions, $\bar{a}$, and all received messages, z_{ji}, to estimate the outgoing message for agent, i, as $\hat{z}_{ij}$. As such, we derive the ELBO that is used for MIM between messages, $\hat{z}_i$, for agent, i, and joint observations, $\bar{o}$, as in Eq. Equation 8.3, where Θ_i is the reverse model for agent i.

$$I(\hat{z}_i;\bar{o}) = H(\hat{z}_i) - H(\hat{z}_i|\bar{o}) \geq \mathbb{E}_{\hat{z}_i\sim\mathcal{N}(\vec{\mu},\vec{\sigma}^2)}\Big[\log\big(\Theta_i(\hat{z}_i|\bar{o})\big)\Big] + H(\hat{z}_i) = L_{MIM}(\pi^i_\phi \parallel \Theta_i) \quad (8.3)$$

Maximizing the evidence lower-bound in Eq. Equation 8.3 is equivalent to maximizing the MI, $I(\hat{z}_i;\bar{o})$.

Training and Execution

We build the MixTURE framework in a Centralized Training for Decentralized Execution (CTDE) paradigm [66] to accommodate for the global joint-action inputs to the MI maximizing reverse models during training. We note that, the MIM reverse model is only used during training and is cut during the execution, and therefore, the learned policies can be executed fully decentralized. To optimize the policies based on the reward signal generated by the discriminators, we leverage the PPO algorithm [304]. Moreover, to enhance and stabilize the training we propose combining an offline BC loss with the online GAIL loss. As shown by prior work [307], through this combination, the offline BC helps preserving ground knowledge that should be respected during training, while the online part helps with learning of new information encountered during execution. As such, putting together our distributed GAIL loss in Eq. Equation 8.1, the PPO loss, the offline BC loss, and the MIM loss introduced in Eq. Equation 8.3, we present the full loss expression to train the MixTURE architecture as follows in Eq. Equation 8.4, where $\mathcal{N}$ is the total number of agents and λ is a tunable scaling parameter.

$$L_{\text{total}} = \sum_{i=1}^{\mathscr{N}} L_{\mathscr{D}_\theta^{(c_i)}} + \lambda_{\text{PPO}} L_{\text{PPO}^{(c_i)}} + \lambda_{\text{BC}} L_{\text{BC}^{(c_i)}} - \lambda_{\text{MIM}} L_{\text{MIM}^{(c_i)}} \tag{8.4}$$

8.1.5 Evaluation

We break the problem of evaluating our proposed architecture for teaching multi-agent coordination policies to (heterogeneous) robot teams into three research questions (RQ):

RQ1 Can MixTURE learn useful multi-agent coordination strategies from synthetic data (e.g, models of human experts / Oz-of-Wizard [308])? Evaluate the quality of learned policies against SOTA baselines and ablations to confirm performance and sample efficiency.

RQ2 Is the MixTURE architecture applicable to learning from real human expert data? Evaluate the performance against baseline with expert demonstrated communication.

RQ3 How challenging is it for a human expert to provide multi-agent demonstration and does MixTURE alleviate the challenge? Compare Workload and System Usability Score (SUS) for when a human subject utilizes MixTURE vs. a classic MA-LfD architecture.

Evaluation Environments

In keeping with prior work in MARL and MA-LfD [110, 13], we selected three multi-agent domains that are partially observable, require collaboration among agents, and include heterogeneous agents (see Section subsection 8.1.3). Please refer to the supplementary material for more details about the environments.

1. **Predator-Prey (PP)** [251]: the goal in this homogeneous (i.e., same class agents)

domain is for $\mathscr{N}$ *predator* agents with limited vision to find a stationary prey and move to its location.

2. **Predator-Capture-Prey (PCP)** [13]: a new class of *capture*-agents, are introduced to the PP. In this heterogeneous domain, the goal of *predator* agents is the same, while *capture* agents must move to the prey location and capture it, without having any observation inputs.

3. **FireComander (FC)** [13]: two classes of robots, *perception* and *action* robots, are required to collaborate to extinguish a propagating firespot. Similar to the PCP domain, *perception* robots search the domain to find hidden firespots, and *action* robots, which do not have any observation inputs, must relay on communication to put out the firespots using an extra action when on a fire. Unlike the PCP, in this complex domain firespots randomly spread over time and thus, the team must continue until all firespots are found and extinguished.

Baselines

To investigate our **RQ1**, we benchmark our method in the three environments introduced in Section subsubsection 8.1.5 against a variety of baselines described below. All baselines utilize the combined offline BC and online loss training scheme described in Section subsubsection 8.1.4. The expert heuristic for environment action is a near-optimal search algorithm and for the communication heuristic it is an anticipatory observation sharing mechanism inspired by prior work [309, 310]. Please refer to the supplementary material for more details on the expert heuristics design for each domain.

- **MARL** [304]: MA-PPO optimizing both environment-action and communication policies .

- **BC+DC** [106]: MixTURE ablation trained only via offline BC loss on synthetic dataset.

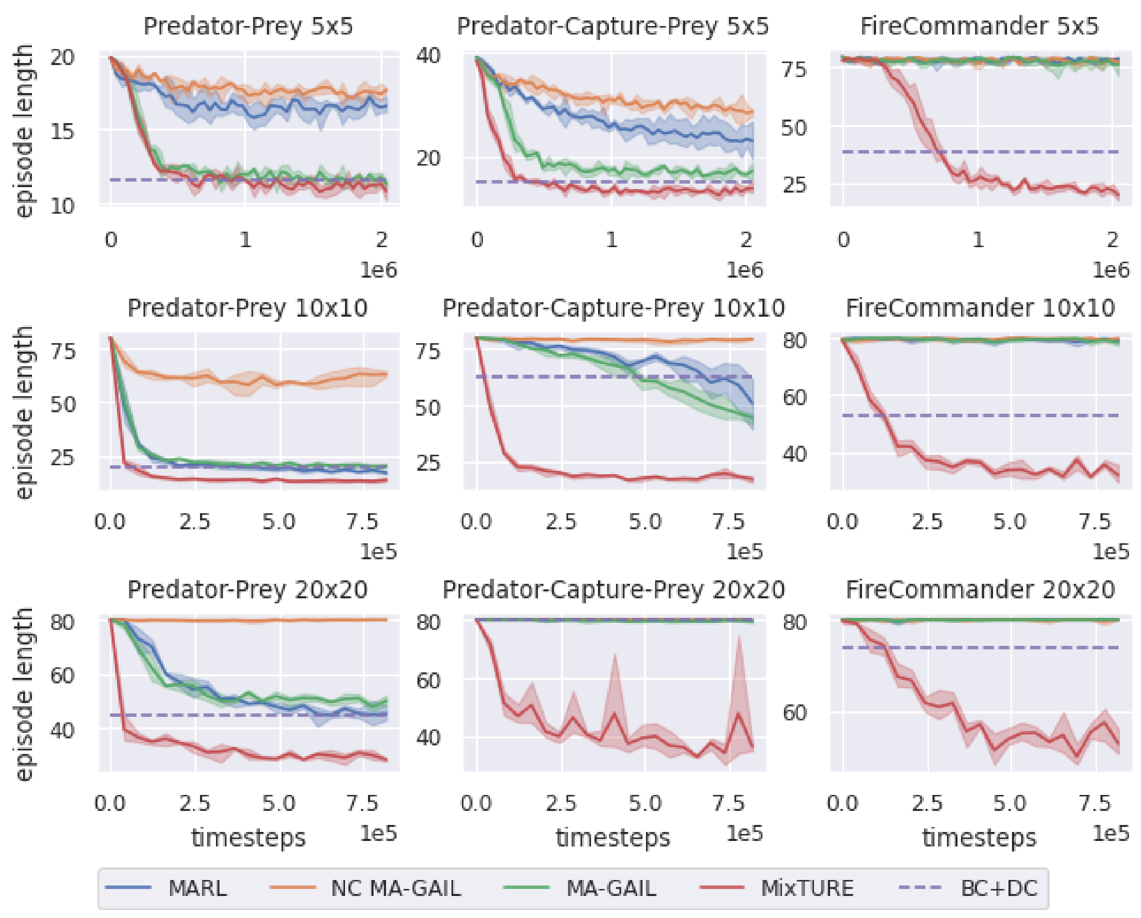


Figure 8.2: Full evaluation results for MixTURE and the baselines in the all difficult levels (i.e., easy, moderate, and hard) of the three environments (i.e., PP, PCP, and FC).

- **NC MA-GAIL** [110]: Non-communicative ablation of the MA-GAIL [110] trained on synthetic data w/o communication (i.e., independent agents).
- **MA-GAIL** [110]: Full MA-GAIL trained on full synthetic dataset w/ communication.

Human-Subject Experiment: Conditions and Procedure

To investigate our **RQ2** and **RQ3**, we conducted an IRB-approved human-subject user study in the FireCommander domain. For more details about the study and the environment, please refer to the supplementary material.

Domain, Setup, and Procedure – We built a heterogeneous multi-agent environment in the FireCommander (FC) domain [13] in which a human expert played different difficulty levels and modes of the FC as a strategic game to provide demonstrations for training the composite robot team. Depending on the level of difficulty (i.e., easy, moderate, hard), there can be multiple initial firespots, hidden from the human, that propagate randomly based on a fixed wall-clock rate. The human expert was responsible to strategically move the simulated robots to find and extinguish all firespots as fast as possible using perception and action agent. The human subject was shown a performance score at the end of each round, computed based on existing, found, and extinguished fires. Each subject began by filling in a pre-questionnaire form followed by reading through a series of detailed game instructions and objectives as well as a few expert tips and guidance. The subject then was allowed to practice different modes of the game. Next, each subject played six different rounds of the game (i.e., two modes/conditions and three difficulty levels). The demonstration data was fully stored for all of the non-practice rounds to be later used for training. Finally, each subject was asked to fill some post-measurement forms.

Participants – We recruited 55 participants in an IRB-approved experiment, whose ages range from 20 to 32 ($25.0\% \pm 2.67$). All participants were recruited through on-university-campus advertisement including 34.5% females. All participants were trained equally for the task through instructions, tips, videos, and practice rounds.

Conditions – In this study, we seek to determine: (1) if MixTURE can learn multi-agent collaborative policies from diverse human generated data w/o demonstrated communication and its performance against classic MA-GAIL w/ expert demonstrated communication, and (2) how does demonstrating versus not demonstrating the communication strategy affect the human expert's performance, workload, and system usability measures. As such, we utilize a 1×2 within-subjects design varying across two abstractions: (1) only demonstrating environment actions for each robot at each time step (i.e., *noComm* condition), 2) demonstrating both environment actions and communication actions for each agent at each time step ((i.e., *withComm* condition)).

Metrics – To evaluate our **RQ2**, the demonstration data collected from the human subjects for both *noComm* and *withComm* conditions are used to train out MixTURE and the classic MA-GAIL [110], respectively. The algorithms are compared in terms of performance (i.e., number of steps taken to find and extinguish all fires where lower is better), KL-Divergence, D_{KL}, between the marginal state distribution in human data and the learned policy rollout distribution (i.e., lower is better) [285], and the log-likelihood, LL, metric as the probability of demonstrated action over given states, using the learned policy (i.e., higher is better) [285]. Additionally, to address our **RQ3**, we leverage the NASA-TLX Workload Survey [311] and the System Usability Scale (SUS) [312] post measurements.

Results and Discussion

Baseline Comparisons on Synthetic Dataset – We evaluated the MixTURE against all baselines (Section subsubsection 8.1.5) in all three domains introduced in Section subsubsection 8.1.5 and under three different difficulty levels: (1) easy (5×5 domain, 3 robots), (2) medium (10×10 domain, 6 robots), and (3) hard (20×20 domain, 10 robots). More environment details are provided in the supplementary material. Table Table 8.1 presents the full evaluation results for MixTURE and baselines including the KL-Divergence (D_{KL}) and Log-Likelihood (LL) metrics. The results show the mean values across metrics calculated over ten trials of running the best models stored during training. As shown, MixTURE achieves significant improvement over all baselines and in all domains. Our evaluations confirm MixTURE's ability to learn highly complex collaborative policies in heterogeneous, partially observable, and dynamic environments. We believe our model provides a strong step towards learning collaborative policies in multi-robot systems by setting a new SOTA in complex heterogeneous tasks.

Fig. **??** shows the training and evaluation results for MixTURE and the baselines in the medium case, for PP, PCP, and FC domains. Each epoch on the x-axis represents $40K$ data samples. As shown, MixTURE outperforms all non-communicative, communicative with expert heuristic, and communicative with differentiable communication channels in learning

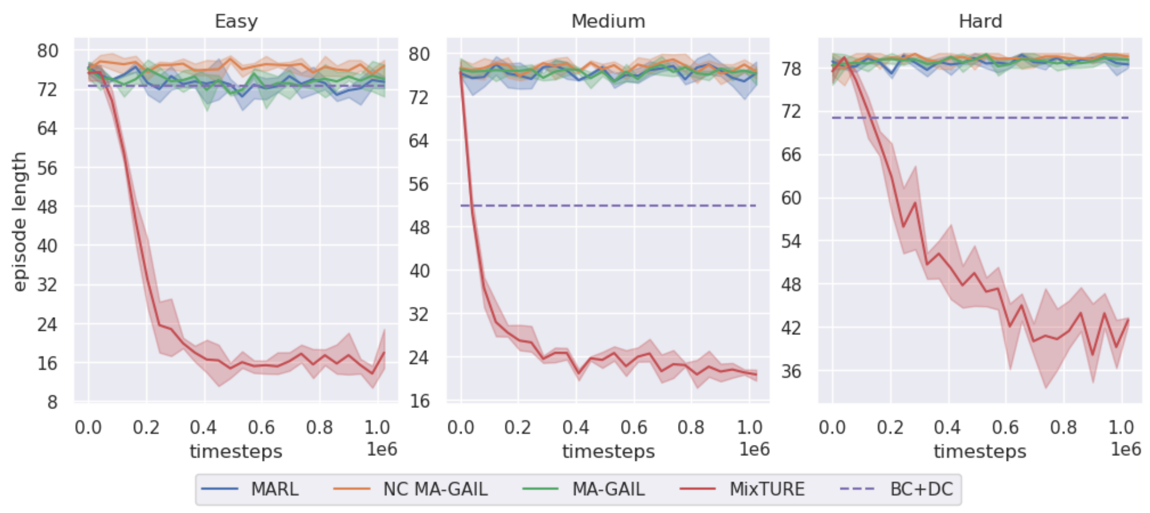


Figure 8.3: Evaluation results for MixTURE and MA-GAIL [110] on real human data. From left to right, the figures present results for easy, moderate, and hard scenarios (see Section subsubsection 8.1.5). MixTURE can learn high-quality multi-agent policies from human demonstration.

collaborative teaming policies. Under the similar training schemes, not only MixTURE outperforms the baselines at the convergence, it also shows significantly lower sample complexity. We note that the medium case of the FC domain was reported to be very challenging in prior work [13] even for MARL methods with differentiable communication channels. As we can see, by combining knowledge acquired from the expert demonstration as well as the introduced MIM-based end-to-end communication, MixTURE can learn a high-quality policy for a heterogeneous team of robots in this challenging domain.

Training and Evaluation Results on Human-Subject Dataset – We train MixTURE and MA-GAIL [110] on data collected in our human-subject user experiment to investigate our **RQ2**. The results are presented in Fig. Figure 8.3. From left to right, the figures present results for easy, moderate, and hard scenarios (see Section subsubsection 8.1.5). As shown, MixTURE can learn high-quality multi-agent collaboration policies from human demonstrations. This is while purely relaying on human demonstrations for both environment- and communication-action strategies, as suggested in prior work [110, 111], does not succeed in this task. We note that, although humans generally tend to be diverse in styles, prefer-

Easy

	Heu.	Diff.	Predator-Prey # Steps	Predator-Prey D_{KL}	Predator-Prey LL	Predator-Capture-Prey # Steps	Predator-Capture-Prey D_{KL}	Predator-Capture-Prey LL	FireCommander # Steps	FireCommander D_{KL}	FireCommander LL
MARL	✓	-	15.57	1.044	-67.54	23.38	1.050	-57.70	78.77	0.871	-36.65
BC+DC	-	✓	11.84	0.919	-0.81	15.38	0.887	-1.44	38.95	1.059	-0.11
NC MA-GAIL	-	-	17.48	1.129	-9.75	27.60	1.070	-8.45	76.49	1.112	-7.60
MA-GAIL	✓	-	11.68	1.074	-4.29	16.43	1.094	-10.08	77.46	1.059	-15.00
MixTURE	-	✓	11.16	0.908	-3.19	13.08	0.850	-3.81	22.31	0.896	-4.58

Medium

	Heu.	Diff.	Predator-Prey # Steps	Predator-Prey D_{KL}	Predator-Prey LL	Predator-Capture-Prey # Steps	Predator-Capture-Prey D_{KL}	Predator-Capture-Prey LL	FireCommander # Steps	FireCommander D_{KL}	FireCommander LL
MARL	✓	-	17.93	1.061	-1.50	56.04	0.964	-3.48	79.09	0.920	-123.09
BC+DC	-	✓	21.28	1.019	-0.35	61.54	0.907	-0.27	44.70	1.060	-12.47
NC MA-GAIL	-	-	60.38	1.086	-2.73	77.77	1.125	-4.43	79.47	1.060	-7.51
MA-GAIL	✓	-	20.39	1.059	-1.84	44.84	1.035	-3.34	79.02	1.062	-104.45
MixTURE	-	✓	13.15	1.023	-0.73	17.27	1.016	-1.59	34.82	1.061	-7.39

Hard

	Heu.	Diff.	Predator-Prey # Steps	Predator-Prey D_{KL}	Predator-Prey LL	Predator-Capture-Prey # Steps	Predator-Capture-Prey D_{KL}	Predator-Capture-Prey LL	FireCommander # Steps	FireCommander D_{KL}	FireCommander LL
MARL	✓	-	47.40	1.057	-2.33	79.76	0.979	-4.09	80.00	0.992	-193.18
BC+DC	-	✓	46.16	1.044	-0.84	79.75	0.953	-2.04	72.87	1.062	-27.69
NC MA-GAIL	-	-	79.52	1.064	-5.03	80.00	1.064	-9.49	79.98	1.061	-12.68
MA-GAIL	✓	-	49.08	1.055	-2.99	79.86	1.036	-6.86	80.00	1.059	-157.58
MixTURE	-	✓	28.73	1.055	-1.06	36.41	1.046	-1.99	56.49	1.063	-9.89

Table 8.1: Full evaluation results for MixTURE and baselines in all domains and all difficulty levels on expert heuristic dataset. Heu. and Diff. indicate the models access to heuristic or differentiable communication, respectively. Bolded values indicate the best models.

ences, and strategies for LfD tasks [285, 306], such diversity and heterogeneity seems to have a minimal effect on the performance of MixTURE. We hypothesize that MixTURE's MIM-based differentiable communication channels provide the model with ability to reason about the underlying human demonstrations and cope with trajectory distribution through automatically finding a suitable communication protocol. We believe that this strong result shows great potential for the MixTURE model in teaching multi-agent coordination and collaboration policies to robot teams through human demonstrations effectively and efficiently.

Statistical Analysis – We investigate the our **RQ3** by quantifying the workload and SUS measures reported by the human subjects. We hypothesize that:

H1 Demonstrating both an environment-action and a communication-action strategy for the robot team increases the human experts workload, measured by NASA-TLX [311].

H2 Demonstrating both an environment-action and a communication-action strategy for the robot team decreases the system's usability scale [312].

H3 Demonstrating both an environment-action and a communication-action strategy for the robot team negatively affects the demonstration quality.

H1: We test for normality and homoscedasticity and do not reject the null hypothesis in either case, using Shapiro-Wilk test ($p > 0.32$ and $p > 0.38$). We perform a paired t-test and find that using the MixTURE model w/o communication demonstration was rated statistically significantly lower than using Multi-Agent Generative Imitation Learning (MA-GAIL) w/ demonstrated communication by the expert ($p < 0.001$) on NASA-TLX workload scale. As shown in Fig. Figure 8.4, relaxing the need for demonstrating a communication strategy reduced the humans' workload by 44.3% in our experiment.

H2: We test for normality and homoscedasticity and do not reject the null hypothesis in either case, using Shapiro-Wilk test ($p > 0.96$ and $p > 0.08$). We perform a paired t-test and find that using the MixTURE model w/o communication demonstration led to a statistically

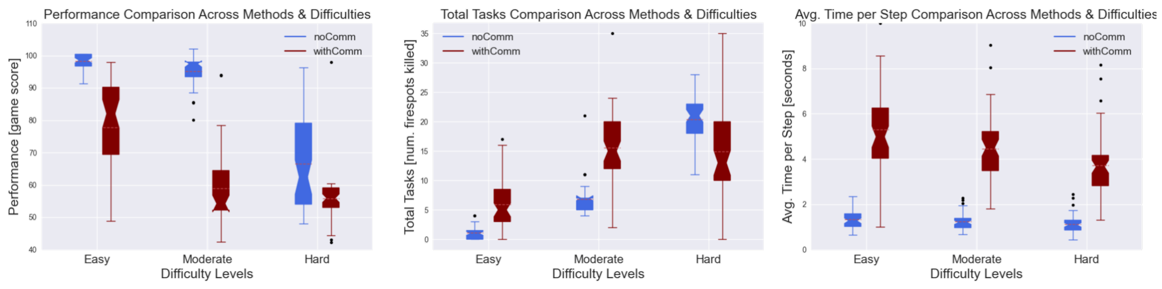


Figure 8.5: Statistical analysis for the human subject data, supporting the **H3**. Results indicate that using the MixTURE model w/o communication demonstration leads to significantly higher performance, more tasks completed and a better ability to scale to the more complex scenarios with more tasks, and significantly lower demonstration time per step.

significantly higher SUS than using MA-GAIL w/ demonstrated communication by the expert ($p < 0.001$). As shown in Fig. Figure 8.4, relaxing the need for demonstrating a communication strategy increased the systems' usability scale by 46.1% in our experiment.

H3: We perform a statistical analysis on: (1) human's performance score in the game, (2) total tasks completed (i.e., fires killed), and (3) average demonstration time per step. We perform a Kruskal–Wallis and confirm that (see Fig. Figure 8.5) in our experiments, relaxing the need for demonstrating a communication strategy through MixTURE leads to significantly higher performance ($p < 0.001$), more tasks completed and a better ability to scale to the more complex scenarios with more tasks ($p < 0.001$), and significantly lower demonstration time per step ($p < 0.001$).

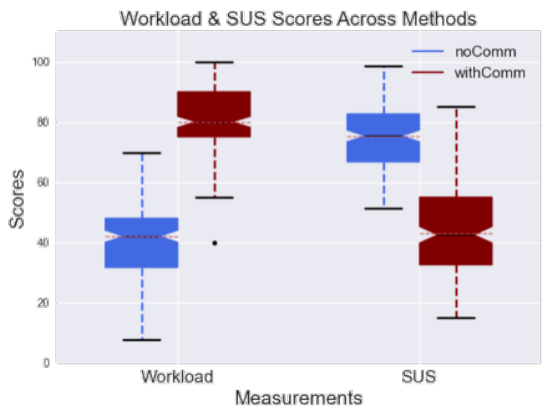


Figure 8.4: Workload and SUS measures, compared for against the experiment conditions.

Limitations and Future Work

A limitation of the proposed MixTURE architecture is that is does not directly attempt to tackle the problem of heterogeneity and different styles in human demonstrations. Demonstrating multi-agent strategies can be considered a highly involved and high-workload task, which in turn can affect a human's situational awareness and optimality of demonstrations. An interesting future direction is then to leverage existing approaches for learning from suboptimal demonstration such as the Self-Supervised Reward Regression (SSRR) [285] to enable MixTURE to learn optimal policies from suboptimal human demonstrations.

8.1.6 Conclusion

We proposed the MixTURE model to learn multi-agent collaborative policies for a robot team, directly from human expert demonstrations. Using our method, a human expert can only teach the robot team how to accomplish a task collaboratively via demonstrations and the team will automatically reason for and learn a communication strategy suitable for the underlying demonstrations. The learned communication protocol will then help the robot team to deal with the partial observability, reasoning about action-decisions to best respond to teammates' policies, and alleviate the effects of environment non-stationarity. We provided several empirical and experimental results, confirming MixTURE's strong ability to learn from expert heuristics and real human generated data and outperform all baselines in several complex domains with heterogeneous robots and tasks.

CHAPTER 9
LIMITATIONS AND FUTURE WORK

Our coordinated planning algorithms, presented in chapter 5, is model-based. Model-based methods can generally be implemented efficiently in real-time, however, their reliance on model accuracy and communication reliability (see chapter 5) makes the performance of such methods vulnerable to uncertainties and disturbances of the system and environment [313]. In our model-based frameworks, UAVs quantify their estimation uncertainty by explicitly inserting the inferred model parameters from the environment into the model and following a nonlinear uncertainty propagation law. According to our experiments reported in section 5.1, a mismatch between the actual model used by the algorithm and the ground truth model can lead to significantly increasing the actual measurement uncertainty residuals and consequently an unreliable coverage plan. While this poses a limitation on the proposed frameworks, in such cases, the best practice is to use adaptive estimation methods (e.g., AEKF). Nevertheless, the results of such methods can still be unreliable [7].

With increased domain complexity and dimensionality, data-driven and learning-based approaches such as MARL can greatly suffer from high sample complexity, dramatic increase of wall-clock training time, and non-generic reward shaping. Particularly, our proposed HetNet model in chapter 6 also demonstrated such problems when trained on larger environments (e.g., 20×20 state space) or more complex domains such as FireCommander. The learned binary communication protocol shared among agents can also susceptible to noisy (e.g., bit flip) and lossy communications. The sample efficiency and wall-clock training time issues can be positively addressed by modifying the training architecture to incorporate multi-processing and parallel computation paradigms. Further, incorporating advanced RL methods such as PPO can help with the quality of learned solutions. We note that we are currently investigating such improvements to the HetNet architecture and an enhanced version of this model will be published in a journal extension soon.

Our InfoPG model for iterated multi-agent decision rationalization, proposed in chapter 7, can also suffer from a few computational limitations. MI maximization by applying the strictly non-negative reward condition in InfoPG objective (Equation 7.3) comes at the cost of zeroing out negative experiences which may have an impact on sample-efficiency. Moreover, for environments with many interacting agents, as the rationalization level, k, increases, the computational overhead of calculating the action-conditional distributions also raises. Lastly, the action-conditional policies in InfoPG's architecture are limited to homogeneous action-spaces.

Lastly, a limitation of our proposed MixTURE architecture, in chapter 8, is that is does not directly attempt to tackle the problem of heterogeneity and different styles in human demonstrations. Demonstrating multi-agent strategies can be considered a highly involved and high-workload task, which in turn can affect a human's situational awareness and optimality of demonstrations. MixTURE also does not directly formulate a solution to address learning from sub-optimal demonstrations. As such, an interesting future direction is to leverage existing approaches for learning from suboptimal demonstration such as the Self-Supervised Reward Regression (SSRR) [285] to enable MixTURE to learn optimal policies from suboptimal human demonstrations. It is also interesting to investigate the effects of added levels of autonomy to the agents in the robot team to a human demonstrator's solution optimality and workload in time-sensitive tasks, such as FireCommander.

CHAPTER 10
CONCLUSION

In this thesis, I presented several contributions towards enabling true teamwork and collaboration for robot teams. By transitioning from single to multi-robot systems, I studied how to leverage interactions between robots to do even more than what one single robot could do on its own. This includes studying higher order goals; goals that can be potentially only be achieved through a collective of autonomous agents.

In chapter 4, we presented two model-based multi-robot coordinated control frameworks. First, in [5] we combined a node-level control criteria and an ensemble-level control criteria to introduce a novel coordinated control algorithm for human-centered active sensing of wildfires, providing high-quality, online information to human firefighters on the ground. In our approach, we took advantage of AEKF's error propagation capability to generate an uncertainty map, incorporating uncertainties about firefront dynamics and areas of human activity. We showed that our approach outperformed prior work for distributed control of UAVs for wildfire tracking as well as a reinforcement learning baseline. Next, we developed an adaptive coordinated control strategy in [207] for the leader-follower systems with uncertain communication network structures. The proposed approach provides a novel coordinated controller to be applied to disconnected and/or non-communicative teams of robots with uncertain communication graph structure for which the conventional consensus-based networked controllers fail. We achieved this result by bypassing the connectedness condition at the cost of a simple centralized model-reference. Simulation and experiments confirm the benefits of the proposed scheme over conventional methods that fail at achieving consensus.

In chapter 5, we built on our works in previous chapter by moving the focus of the algorithm design from the low-level control input to the high-level decision-making and planning under uncertainty. First, in [7, 8] we introduced a novel analytical measurement-residual bound on fire propagation uncertainty, allowing high-quality planning under environment uncertainty for real-time wildfire monitoring and tracking, while also providing a probabilistic guarantee on the quality of service. Our approach outperformed prior work for distributed control of UAVs for wildfire tracking, as well as a reinforcement learning baseline. Physical implementation of our framework on real robots in a multi-robot testbed demonstrated and validated the feasibility of our approaches. Next, we extended this work to include heterogeneous robots (i.e., robots with different observation- and action-spaces as well as different tasks and objectives). In [9], we introduced a novel hierarchical approach to tackle the high-level decision making and low-level collaborative control problems for heterogeneous teams of autonomous robots consisting of perception agents and action agents. In our centralized high-level decision-making module, we proposed MA-SARTSA-based learning under a MA-POSMDP model to enable perception agents to explore an unknown environment (i.e., discover dynamic targets) and exploit known targets by extracting their state information. We also introduced a measurement-uncertainty based tracking error and derived a set of analytical upper-bound service times to ensure a probabilistically guaranteed service for action agents in various scenarios. Additionally, we introduced a coordinated routing problem with an attribute-based robot-interaction scheme through which the perception-action agents cooperation is individualized to account for robots heterogeneity and improve the composite team's resiliency and performance.

In chapter 6, we moved beyond model-based approaches and incorporated data-driven methods, such as MARL to address the challenges in model-based methods such as model inaccuracies and failure. In [13, 10], motivated by the diverse communication patterns across collaborating human teams, we presented a communicative, cooperative MARL framework for learning heterogeneous cooperation policies among agents of a *composite* team [54, 121]. We proposed Heterogeneous Policy Network (HetNet), a heterogeneous graph-attention based architecture, and introduced the Multi-Agent Heterogeneous Actor-

Critic (MAHAC) learning paradigm for training HetNet to learn class-wise cooperation policies. We pushed the boundaries beyond performance considerations as in prior work by equipping HetNet with a binarized encoder-decoder communication channel to facilitate learning a new and highly efficient encoded language for heterogeneous communication. We empirically showed HetNet's superior performance against several baselines in learning both homogeneous and heterogeneous cooperative policies. We provided empirical evidence that show: (1) our binarized model achieves more than 200$\times$ reduction in communication overhead (i.e., message bits) per round of communication while also outperforming baselines in performance, (2) HetNet is robust to varying bandwidth limitations and team compositions.

In addition to communication, individuals in high-performing human teams also benefit from the theory of mind and making strategic decisions by recursively reasoning about the actions (strategies) of other human members [198]. Such hierarchical rationalization alongside with communication facilitate meaningful and strategic cooperation in human teams.

As such, in chapter 7, we proposed a novel information-theoretic, fully-decentralized cooperative MARL framework, called Informational Policy Gradient (InfoPG) [12], where agents iteratively rationalize their action-decisions based on their teammates' actions. We studied cooperative MARL under the assumption of bounded rational agents and leveraged action-conditional policies into policy gradient objective to accommodate our assumption. We leveraged iterated k-level reasoning from cognitive hierarchy theory and presented a collaborative, fully-decentralized MARL framework which explicitly maximizes MI among cooperating agents by equipping each agent with an action-conditional policy and facilitating iterated inter-agent communication for hierarchical rationalizability of action-decisions. We analytically showed that the design of our MI-based PG method, increases an MI lower-bound, which coincides with improved cooperativity among agents. We empirically showed InfoPG's superior performance against various baselines in learning cooperative policies. Finally, we demonstrated that InfoPG's regulatory effect on MI makes it Byzantine-resilient and capable of solving BGPs in fully-decentralized settings.

Finally, in chapter 8, we presented a MA-LfD architecture to learn heterogeneous teaming strategies directly from human experts with domain knowledge. MARL suffers from several problems, including designing an expressive reward function [286] and are typically hard to train with increased domain complexity [285]. As such, in the previous chapter, we proposed the MixTURE model to learn multi-agent collaborative policies for a robot team, directly from human expert demonstrations. Using our method, a human expert can only teach the robot team how to accomplish a task collaboratively via demonstrations and the team will automatically reason for and learn a communication strategy suitable for the underlying demonstrations. The learned communication protocol will then help the robot team to deal with the partial observability, reasoning about action-decisions to best respond to teammates' policies, and alleviate the effects of environment non-stationarity. We provided several empirical and experimental results, confirming MixTURE's strong ability to learn from expert heuristics and real human generated data and outperform all baselines in several complex domains with heterogeneous robots and tasks.

Appendices

APPENDIX A
TIME INDEPENDENCY OF THE EKF'S MEASUREMENT RESIDUAL

Our analytical URR bound in Equation 5.6 depends on the state-estimation measurement residual computed at different time-steps. To maintain control over the measurement uncertainty, we posit that the UAV observers would want the measurement uncertainty residual with respect to a target on the ground not to increase from $t = t_0$ to $t = t_0 + kT_{UB}$ for any positive integer constant k if the UAV observes the target from the same relative position. Therefore, we examine the time-dependency of the propagated error through our EKF formulation. To this end, we follow the mathematical proof and discussions provided in [9] and [5, 8]. We state that the measurement uncertainty about the states of a dynamic point q_t, observed by a flying UAV is independent of time and is only a function of distance between the observer and the point. In the following, we mathematically proof this point.

First, we present how the uncertainty residual is quantified by an EKF. The total uncertainty residual propagated by EKF is composed of a model and an observation measurement uncertainties, both of which follow the general nonlinear uncertainty propagation law, shown in Equation A.1-Equation A.2, where $\Sigma_{t|t-1}$ is the predicted covariance estimate, $\Lambda_{t|t}$ is the innovation (or residual) covariance, F_t and H_t are the process and observation Jacobian matrices, and Q_t and Γ_t are the process and observation noise covariances, respectively.

$$\Sigma_{t|t-1} = F_t \Sigma_{t-1|t-1} F_t^T + Q_t \tag{A.1}$$

$$\Lambda_{t|t} = H_t \Sigma_{t|t-1} H_t^T + \Gamma_t \tag{A.2}$$

Considering Equation A.1-Equation A.2, the gradients in the process, F_t, and observation, H_t, Jacobian matrices are responsible for alterations in the uncertainty values. To compute these gradients, we calculate the derivatives of fire's propagation model, $\mathscr{M}_t$, and UAV's observation model, $\mathscr{O}_t$, with respect to the state variables. As discussed in subsubsection 5.1.4 and considering the introduced state vectors, we first derive the process and observation Jacobian matrices (F_t and H_t) as follows in Equation A.3-Equation A.4, respectively. In Equation A.3-Equation A.4, $t' = t-1$.

$$\left.\frac{\partial \mathscr{M}_t}{\partial \mathscr{S}_i}\right|_{\hat{\Theta}_{t|t'}} = \begin{array}{c} \\ q_t^x \\ q_t^y \\ p_t^x \\ p_t^y \\ p_t^z \\ R_t \\ U_t \\ \theta_t \end{array} \begin{array}{c} \begin{array}{cccccccc} q_{t'}^x & q_{t'}^y & p_{t'}^x & p_{t'}^y & p_{t'}^z & R_{t'} & U_{t'} & \theta_{t'} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \frac{\partial q_t^x}{\partial R_{t'}} & \frac{\partial q_t^x}{\partial U_{t'}} & \frac{\partial q_t^x}{\partial \theta_{t'}} \\ 0 & 1 & 0 & 0 & 0 & \frac{\partial q_t^y}{\partial R_{t'}} & \frac{\partial q_t^y}{\partial U_{t'}} & \frac{\partial q_t^y}{\partial \theta_{t'}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{A.3}$$

$$\left.\frac{\partial \mathscr{O}_t}{\partial \Phi_i}\right|_{\hat{\Phi}_t} = \begin{array}{c} \\ \varphi_t^x \\ \varphi_t^y \\ \hat{R}_t \\ \hat{U}_t \\ \hat{\theta}_t \end{array} \begin{array}{c} \begin{array}{cccccccc} q_t^x & q_t^y & p_t^x & p_t^y & p_t^z & R_t & U_t & \theta_t \end{array} \\ \begin{pmatrix} \frac{\partial \varphi_t^x}{\partial q_t^x} & \frac{\partial \varphi_t^x}{\partial q_t^y} & \frac{\partial \varphi_t^x}{\partial p_t^x} & \frac{\partial \varphi_t^x}{\partial p_t^y} & \frac{\partial \varphi_t^x}{\partial p_t^z} & 0 & 0 & 0 \\ \frac{\partial \varphi_t^y}{\partial q_t^x} & \frac{\partial \varphi_t^y}{\partial q_t^y} & \frac{\partial \varphi_t^y}{\partial p_t^x} & \frac{\partial \varphi_t^y}{\partial p_t^y} & \frac{\partial \varphi_t^y}{\partial p_t^y} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{A.4}$$

In Equation A.3-Equation A.4, we define the process state vector as $\vec{\Theta}_t = \left[q_t^x, q_t^y, p_t^x, p_t^y, p_t^z, R_t, U_t, \theta_t\right]^T$ and $\vec{\Phi}_t = \left[\varphi_t^x, \varphi_t^y, \hat{R}_t, \hat{U}_t, \hat{\theta}_t\right]^T$ as the mapping vector. As such, we calculate the partial derivatives in Equation A.3 by using Equation 3.1-Equation 3.2 and applying the chain-rule to compute the derivatives of q_t^x and q_t^y with respect to parameters R_{t-1}, U_{t-1}, and θ_{t-1}. The partial derivatives are then derived as in Equation A.5-Equation A.7, where $\mathscr{D}(\theta)$ is $\sin\theta$ and $\cos\theta$ for X and Y axis, respectively.

$$\frac{\partial q_t}{\partial \theta_{t-1}} = C(R_t, U_t)\frac{\partial \mathscr{D}(\theta)}{\partial \theta}\delta t \tag{A.5}$$

$$\frac{\partial q_t}{\partial R_{t-1}} = \left(1 - \frac{LB(U_t)}{LB(U_t) + \sqrt{GB(U_t)}}\right)\mathscr{D}(\theta)\delta t \tag{A.6}$$

$$\frac{\partial q_t}{\partial U_{t-1}} = \frac{R_{t'}\left(LB(U_{t'})\frac{\partial GB(U_{t'})}{\partial U_{t'}} - GB(U_{t'})\frac{\partial LB(U_{t'})}{\partial U_{t'}}\right)}{\left(LB(U_{t'}) + \sqrt{GB(U_{t'})}\right)^2}\mathscr{D}(\theta)\delta t \tag{A.7}$$

To compute the partial derivatives in the observation Jacobian matrix in Equation A.4, we first need to derive the relation between the angle parameters, φ_t^x and φ_t^y, and the UAV pose. The angle parameters contain information regarding both firefront location $[q_t^x, q_t^y]$ and UAV coordinates $[p_t^x, p_t^y, p_t^z]$. According to Figure 4.3, by projecting the looking vector of UAV to planar coordinates, the angle parameters are calculated as shown in EquationsEquation A.8-Equation A.9 for X and Y axes respectively, where $q_t = [q_t^x, q_t^y]$ and $p_t = [p_t^x, p_t^y]$.

$$\varphi_t^x = \tan^{-1}\left(\frac{p_t^z}{\|q_t - p_t\|}\right) \tag{A.8}$$

$$\varphi_t^y = \tan^{-1}\left(\frac{\|q_t - p_t\|}{p_t^z}\right) \tag{A.9}$$

The partial derivatives in the observation Jacobian matrix H_t for X-axis, presented in Equation A.4, are derived as in Equation A.10-Equation A.12 and for Y-axis derivatives, we can derive as in Equation A.13-Equation A.15.

$$\nabla_{q_t}\varphi_t^x = \frac{1}{1+\left(\frac{p_t^z}{\|q_t-p_t\|}\right)^2}\left(\frac{-p_t^z(q_t-p_t)}{\|q_t-p_t\|^3}\right) = \left[\frac{\partial\varphi_t^x}{\partial q_t^x}, \frac{\partial\varphi_t^x}{\partial q_t^y}\right] \tag{A.10}$$

$$\nabla_{p_t}\varphi_t^x = \frac{1}{1+\left(\frac{p_t^z}{\|q_t-p_t\|}\right)^2}\left(\frac{p_t^z(q_t-p_t)}{\|q_t-p_t\|^3}\right) = \left[\frac{\partial\varphi_t^x}{\partial p_t^x}, \frac{\partial\varphi_t^x}{\partial p_t^y}\right] \tag{A.11}$$

$$\frac{\partial\varphi_t^x}{\partial p_t^z} = \frac{1}{1+\left(\frac{p_t^z}{\|q_t-p_t\|}\right)^2}\left(\frac{1}{\|q_t-p_t\|}\right) \tag{A.12}$$

$$\nabla_{q_t}\varphi_t^y = \frac{1}{1+\left(\frac{\|q_t-p_t\|}{p_t^z}\right)^2}\left(\frac{(q_t-p_t)}{p_t^z\|q_t-p_t\|}\right) = \left[\frac{\partial\varphi_t^y}{\partial q_t^x}, \frac{\partial\varphi_t^y}{\partial q_t^y}\right] \tag{A.13}$$

$$\nabla_{p_t}\varphi_t^y = \frac{1}{1+\left(\frac{\|q_t-p_t\|}{p_t^z}\right)^2}\left(\frac{-(q_t-p_t)}{p_t^z\|q_t-p_t\|}\right) = \left[\frac{\partial\varphi_t^y}{\partial p_t^x}, \frac{\partial\varphi_t^y}{\partial p_t^y}\right] \tag{A.14}$$

$$\frac{\partial\varphi_t^y}{\partial p_t^z} = \frac{1}{1+\left(\frac{\|q_t-p_t\|}{p_t^z}\right)^2}\left(\frac{-\|q_t-p_t\|}{(p_t^z)^2}\right) \tag{A.15}$$

Now, considering EKF's covariance propagation Equations in Equation A.1-Equation A.2 as well as the gradients in process Jacobian matrix F_t as calculated in Equation A.5-Equation A.7, we can see that the gradients in process Jacobian matrix are only functions of fire propagation model parameters (e.g., the FARSITE model in this case) such as fuel coefficient, R_t and wind velocity and direction, U_t and θ_t. Consequently, while these parameters do not vary significantly with time, the uncertainty drop due to process model is time-invariant. We note that FARSITE [122] assumes locality in time (i.e., within seconds or few minutes), making the assumption of time-invariant fire parameters fairly acceptable [149]. Moreover, the gradients in the observation Jacobian matrix, Equation A.10-Equation A.15, are only functions of the Euclidean distance between the UAV pose and firespot coordinates. We also know that, since at the time of visiting a firespot the planar displacement between UAV and fire locations are approximately zero and the only distance between the two equals to the UAV altitude. Accordingly, both F_t and H_t are locally time-invariant and the total measurement uncertainty residual variations between two different time-steps (e.g., $t = t_0$ and $t = t_0 + kT_{UB}$) is not a function of time and is only a function of the UAV observer's altitude.

APPENDIX B
CALCULATING COORDINATES OF REACHABLE POLYGON'S VERTICES IN SCANNING FRAMEWORK

Figure B.1 is presented to elaborate on the calculation of coordinates enclosing the reachable areas by the manipulator agent. To calculate the X-Y coordinates of the four vertices, shown in Figure B.1, we first need to form the reachable polygon. As such, we draw two circles centered at the agent's current position (green arches in Figure B.1) with radius $\mathscr{D}_{min} = v_{min}^{M}\delta$ and $\mathscr{D}_{max} = v_{max}^{M}\delta t$. $\mathscr{D}_{min}$ and $\mathscr{D}_{max}$ are the agent's lower and upper-bound planar displacement for one unit of time, δt, if the agent is moving with its minimum or maximum velocity. We call these two circles, the *planar displacement circles*. Next, we need to account for the action agent's angular velocity, ω_{max} (e.g., turning bank). Accordingly, we draw two more circles (blue dashed-circles in Figure B.1) from the agent's current location in clockwise and counter-clockwise directions. We refer to these two circles as *angular motion circles*. The area enclosed by the intersections of these four circles is the *reachable* area by the action agent, M, with velocity range of $\left[v_{min}^{M}, v_{max}^{M}\right]$ and maximum turning bank of ω_{max}.

Accordingly, the desired coordinates of the four vertices of the reachable polygon can be obtained by projecting the agent's current XY coordinates according to the joint angular-planar rotation mappings, as described in subsubsection 5.2.4. In Equation 5.39, $(-1)^i$ is to account for the clockwise and counter-clockwise directions of the angular velocity, ω_{max}, and $R_z^{\varphi}(\varphi)$ is the rotation matrix around the z-axis as in Equation B.1.

$$R_z^{\varphi}(\varphi) = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{bmatrix} \tag{B.1}$$

To calculate the angle, φ, according to Figure B.1, we consider the velocity-vector

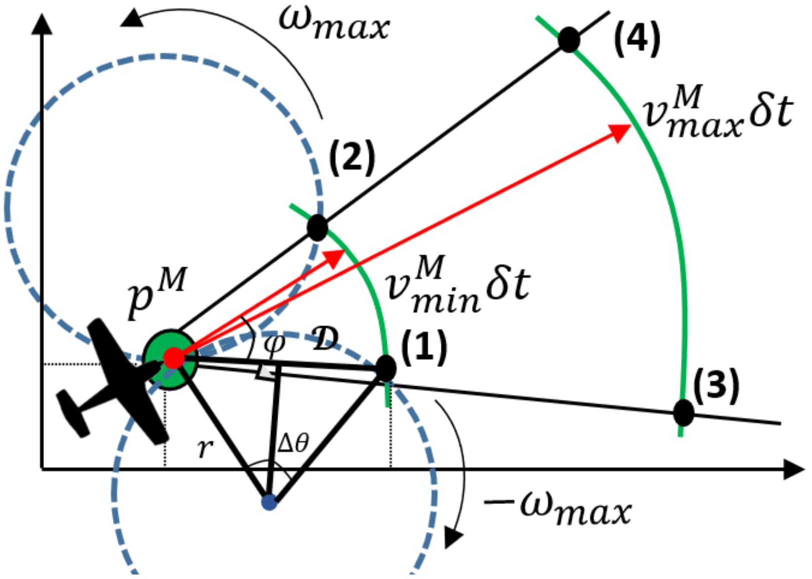


Figure B.1: Vertex coordinates of the reachable polygon, introduced in Figure 5.13, are calculated to generate fixed-wing-friendly paths.

tangent to the angular motion circle. The planar position rotation angle $\Delta\theta$ can be calculated as $\Delta\theta = \omega_{max}\delta t$, and therefore, the angle between the tangent velocity vector and planar displacement line, φ, can be calculated as $\varphi = \frac{\Delta\theta}{2} = \frac{\omega_{max}\delta t}{2}$. The proposed scanning framework is of particular interest for action robots that have restricted motions (i.e., fixed-wing UAVs with non-zero minimum velocity, $v_{min} \neq 0$, and $\omega \in [-\omega_{max}, \omega_{max}]$). In such cases, our coordinated scanning framework leads to action agents being able to directly traverse between any two points inside the reachable polygons with near-linear approximation. In the case of omni-directional robots (e.g., multi-rotor UAVs), the proposed scanning approach can be ignored.

APPENDIX C
SUPPLEMENTARY INFORMATION (chapter 6)

C.1 Evaluation Environment: Additional Details and Parameters

Here we provide additional details regarding the employed evaluation environments for training and testing InfoPG and cover the associated environment parameters related to our experiments. For reproducibility, we publicly provide our code at `github.com/HetNet`[1].

C.1.1 Predator-Prey (PP) [251]

The objective within this homogeneous environment is for *N* predator agents with limited vision to find a stationary prey and move to its location. The agents in this domain are homogeneous in their state, observation, and action spaces and thus, all agents are of the same *class*. All agents are able to sense/observe the environment and each agent's observation is a concatenated array of the state vectors of all grids within the agent's Field of View (FOV). The predator agents' action-space is of dimension five, including cardinal movements and a null action, and is the same for all agents. Each predator agent will receive a small penalty per timestep until it has discovered the prey. A higher-performing algorithm in this domain is defined as one that minimizes the average number of steps taken by agents to complete an episode.

Within our evaluation, we evaluate in a grid size of 5x5 with 3 predators. We set the maximum steps for an episode to be 80. For the reward, each agent receives -0.05 per time step before they find the prey. An episode is considered unsuccessful if the prey is not discovered within the maximum steps.

[1]Available online at: `https://github.com/CORE-Robotics-Lab/HetNet`

C.1.2 Predator-Capture-Prey (PCP)

In our second domain, we have two classes of agents: *predator* agents and *capture* agents. The first class of agent, called the *predator* agents, have the goal of discovering the prey and have an action-space of dimension five, including cardinal movements and a null (stay) action. *Predator* agents have an observation space similar to the agents in PP domain. The second class of agents, called the *capture* agents, have the objective of locating the prey *and* capturing it. Capture agents differ from the predator agents in both their observation and their action spaces. Capture agents do not receive any observation inputs from the environment (i.e., no scanning sensors) and have an additional action of *capture-prey* in their action-space. This additional action must be used at a prey's location to capture the prey. Note that this domain is an explicit example of the perception-action composite teams. An episode is deemed successful once all agents have completed their class-specific objectives. Each *predator* agent is penalized with -0.05 reward every timestep until it has discovered the prey. Each *capture* agent is also penalized with -0.05 every timestep until it has captured (i.e., find the prey and then capture it) the prey. Note the difference in reward scheme, a *capture* agent may have discovered the prey but will receive a negative reward until the *capture-action* is utilized.

We utilize PCP as a testbed for several test heterogeneous interactions. In our head-to-head evaluation against baselines, we utilize a problem with two *predator* agents and one *capture* agent within a 5x5 grid. We set the maximum steps for an episode to be 80.

C.1.3 FireCommander (FC) [54]

We also evaluate the performance both our HetNet variants, HetNet-Binary and HetNet-Real, in a new cooperative multi-agent environment with heterogeneous agents, called FireCommander (FC) [54, 9]. FireCommander can be categorized as a strategic game, in which a composite team of robots (i.e., UAVs) must collaboratively find hidden areas of propagating wildfire and extinguish the fire in such areas as fast as possible. The robot team in FC is composed of two classes of agents: (1) *perception* agents (class P), which can only sense the environment and detect areas of fire and, (2) *action* agents (class A), which can only manipulate the environment by extinguishing a firespot which has already been detected by class P agents. Neither class P, nor class A agents are capable of accomplishing the task on their own, and therefore must communicate and collaborate.

Under the notations in our problem formulation in section 6.1, we have $\mathscr{C} = \{\mathrm{P}, \mathrm{A}\}$ where, $\mathscr{A}^{(\mathrm{P})} = \{1, 2, \cdots, 4\}$ representing the four primitive motions and $\mathscr{A}^{(\mathrm{A})} = \{1, \cdots, 5\}$, representing the four primitive motions and an extra action which corresponds to extinguishing fire by dousing water. Agents of class P are equipped with fire detection sensors and can observe the environment, receiving an input vector of length 29 for each grid within their FOV. Agents of class A, do not receive any observation from the environment. The reward scheme in this domain includes a small temporal penalty of -0.1 per timestep for all agents, a false water-drop penalty of -0.1 for *action* agents, a -0.1 penalty per new firespot for all agents, and a positive reward of +10 for all agents per each extinguished firespot.

In our head-to-head evaluation against baselines, we utilize a problem with two *perception* agents and one *action* agent within a 5x5 grid and one initial firespot that propagates to a new location at each timestep, leaving the previous grid on fire. We set the maximum steps for an episode to be 300. An episode of the game is marked as successful only if all the active firespots within the map are discovered and extinguished. Please refer to the provided FireCommander supplementary document for further details.

C.2 Supplementary Results and Ablation Studies

In this section, we provide our supplementary results. We first provide the implementation and model details for our experiments and then, present the results of an ablation study on the binarization process for digitizing the communication messages. For reproducibility, we publicly provide our code at github.com/HetNet[2].

[2]Available online at: https://github.com/CORE-Robotics-Lab/HetNet

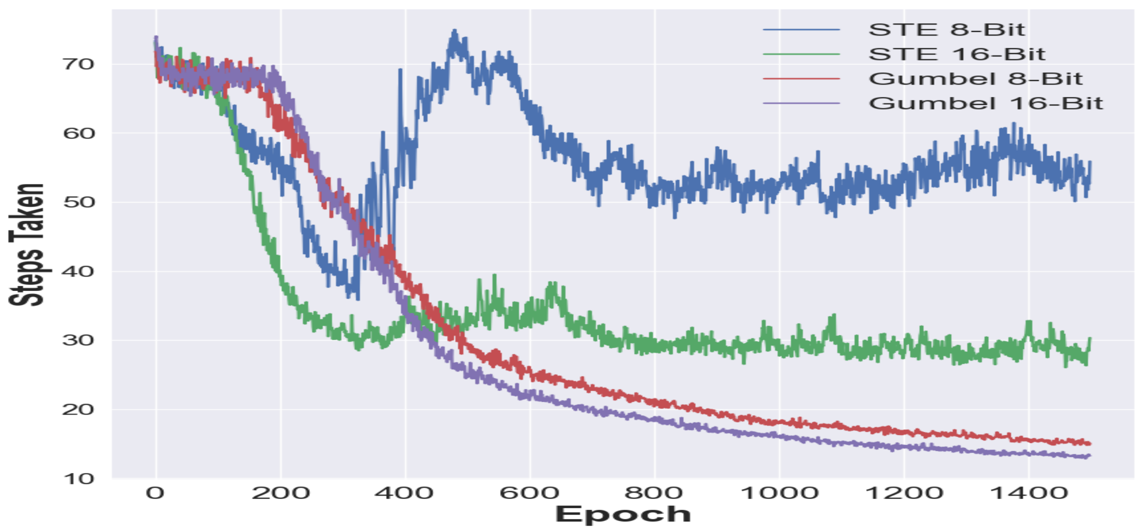


Figure C.1: Performance comparison for two different binarization methods: (1) STE and (2) Gumbel-Softmax, for 8-bits and 16-bits message dimensions.

C.2.1 Implementation and Model Details

For our empirical results, our HetNet implementation consists of three multi-head HetGAT layers stacked on top of the feature preprocessing modules. The first two multi-head layers use $L = 4$ attention heads computing 16 features each (for a total of 64 features merged by concatenation). The final layer also uses $K = 4$ attention heads, but the output dimension is set to the size of an agent's action-space and is merged by averaging. We used the Adam optimizer [314] through training with a learning rate of 10^{-3} for all our experiments and results presented here. We leverage the per-class critic architecture (chosen as a result of a sensitivity analyses detailed in 6.3.5) for all our experiments in Section 6. Algorithm 1 provides a pseudocode to train HetNet with the per-class critic architecture. The policy and critic network parameters are initialized per class (line 2) and the communication steps through HetGAT layers are performed at each step of an episode (line 8). The rest of training procedure implements an on-policy advantage AC procedure except that the gradient updates are class-specific (lines 15-20). We implement HetNet using PyTorch [315] and Deep Graph Library [316]. All of our experiments are performed across three random seeds (0, 1 and 2) and the presented results are averaged across all seeds.

C.2.2 Message Binarization Method: An Ablation Study

Learning a communication model that sends binary messages requires a differentiable approach to convert data from continuous-scale to a discrete (binarized) representation. We experiment with two different techniques that enable binarization: Straight-Through Estimators (STE) with a fixed threshold function [317] and Gumbel-Softmax [264]. The STE binarizes the activations of a layer during forward-pass and directly passes gradients similar to the identity function during backpropagation [318, 319, 319, 320, 321, 322]. The Gumbel-Softmax utilizes a differentiable sampling approach to produce binarized messages from a continuous distribution via categorical reparameterization. By analyzing the results of HetNet with each approach across multiple message dimension sizes (8 and 16 bit), we conclude that Gumbel-Softmax allows for better performance.

APPENDIX D
SUPPLEMENTARY INFORMATION (chapter 7)

D.1 InfoPG Pseudocode

Here, we provide a pseudocode to train our MI maximizing PG algorithm, InfoPG, in algorithm 6.

Consider a MAF-Dec-POMDP introduced in section 7.1, with N agents where each agent is equipped with an *encoding* and a *communicative* policy (π_{enc} and π_{com}, respectively), such that $\pi_{tot} = [\pi_{enc}, \pi_{com}]$ (lines 1-2). At the beginning of a new rollout, and for each timestep, t, within the allowed maximum number of steps, max_cycles, each agent i receives a state observation o_t^i from the environment and produces an initial "guess" action, $a_t^{i,(0)}$ using its encoding policy (lines 5-8). Each agent i has a neighborhood of agents it can communicate with, shown with $j \in \Delta_t^i$ where $|\Delta_t^i|$ is the number of agent i's physical neighbors (i.e., within close proximity). Accordingly, agents obtain the list of their neighbors by using the adjacency graph, $\mathcal{G}_t$ (line 11).

Next, depending on the specified level of iterated rationalizability in the decision hierarchy, K, agents communicate their action guesses as higher-dimensional latent vector distributions with their neighbors K times and update their action guess iteratively using their communicative policy (line 9-14). The level-k action is then executed by all agents and a local reward is given to each agent separately (line 15). This process continues until either the environment is solved successfully, or the allowed maximum number of steps, max_cycles, has been attained. For each timestep, t, of the policy rollout and for each agent i, first the advantage value, A_t^i, is computed using the critic network (line 17) and then, the gradient of the log probability is computed and scaled by the instantaneous advantage (line 18). Note that the ReLU function in line 18 is proposed to enforce the

Algorithm 6: Training the Mutual Information Maximizing Policy Gradient (InfoPG).

```
1: Input: Number of agents, $\mathcal{N}$, $max\_cycles$ and agents' level of iterated rationalizability, $K$
2: Initialize: For all agents $\{\pi_{tot}^1, \pi_{tot}^2, \cdots, \pi_{tot}^{\mathcal{N}}\}$ and $\{V^1, V^2, \cdots, V^{\mathcal{N}}\}$
3: while not converged do
4:   for $t = 1$ to $max\_cycles$ do
5:     Reset environment and receive initial observation set: $\{o_t^1, o_t^2, ..., o_t^{\mathcal{N}}\}$
6:     for $i = 1$ to $\mathcal{N}$ do
7:       Sample initial "guess" action: $a_t^{i,(0)} \sim \pi_{enc}^i(* \mid o_t^i)$, where $\pi_{enc}^i \in \pi_{tot}^i$
8:     end for
9:     for $k = 1$ to $K$ do
10:      for $i = 1$ to $\mathcal{N}$ do
11:        Identify neighbors by obtaining neighbor list, $j \in \Delta_t^i$, using the adjacency graph, $\mathcal{G}_t$
12:        Sample MAP: $a_t^{i,(k)} \sim \pi_{com}^i(* \mid a_t^{i,(k-1)}, \{a_t^{j,(k-1)} \mid j \in \Delta_t^i\})$, where $\pi_{com}^i \in \pi_{tot}^i$
13:      end for
14:    end for
15:    Step through environment using $\{a_t^{1,(k)}, a_t^{2,(k)}, \cdots, a_t^{\mathcal{N},(k)}\}$, and receive next states and rewards: $\{o_{t+1}^1, o_{t+1}^2, ..., o_{t+1}^{\mathcal{N}}\}$, $\{r_t^1, r_t^2, ..., r_t^{\mathcal{N}}\}$
16:    for $i = 1$ to $\mathcal{N}$ do
17:      $A_t^i = r_t^i + V^i(o_{t+1}^i) - V^i(o_t^i)$
18:      $\nabla \pi_{tot}^i = \mathrm{ReLU}(A_t^i) \nabla \log(\pi_{tot}^i(a_t^{i,(k)} \mid \cdots))$   % For Adv. InfoPG remove the ReLU
19:      $\nabla V^i = (A_t^i)^2$
20:      Update: $\pi_{tot}^i = \pi_{tot}^i + \eta \nabla \pi_{tot}^i$
21:      Update: $V^i = V^i + \eta \nabla V^i$
22:    end for
23:    $\{o_t^1, o_t^2, ..., o_t^{\mathcal{N}}\} = \{o_{t+1}^1, o_{t+1}^2, ..., o_{t+1}^{\mathcal{N}}\}$
24:  end for
25: end while=0
```

non-negative reward feedback condition in InfoPG, Equation 7.3. However, other mechanisms could achieve the same effect (e.g., shaping the reward function to only include positive values). Line 19 is showing the gradient of our value estimate. The loss for our value estimate $V(s_t)$ is the cumulative discounted rewards subtracted by the true value of s_t, which is the advantage (the TD error). Therefore, the gradient in line 19 is the sum of squared individual advantages. Next, the encoding and communicative policies are updated (line 20) and eventually, the critic network is updated (line 21). This process repeats until convergence of the cumulative discounted rewards across all agents. We provide our code at https://github.com/CORE-Robotics-Lab/InfoPG

D.2 InfoPG for Continuous Action-Spaces

To deploy InfoPG in continuous action-spaces, we follow common practice for continuous action-space actor-critic methods. Continuous policies are presented as probability distributions with floating values in a certain range (e.g., $(-1,1)$). In this case, in order to facilitate exploration we will sample agents' actions from a normal probability distribution. As such, in continuous action-spaces, the policy network (Actor) will normally have two output heads, instead of one. The two outputs of the policy network are μ and σ, the mean and standard deviation (STD) of the probability distribution, respectively. The sampled actions will be centered around the μ and the σ determines on average, how far from the center the sample values will be. As the network gets more certain, the σ gets smaller, meaning that we tend to exploit good actions rather than exploring.

In discrete action-spaces the loss-function was based on the log-probability (Equation 7.3). In continuous action-spaces, the log-probability of a normal distribution is used, as shown in Equation D.1.

$$log_{\pi_\theta}(a|s) = -\frac{(x-\mu)^2}{2\sigma^2} - log\sqrt{2\pi\sigma^2} \tag{D.1}$$

In Equation D.1, the first and second term are the negative log-probability and the entropy bonus, respectively. By substituting the log-probability of the normal distribution in Equation D.1 back into the original InfoPG objective in Equation 7.2, we can directly deploy the InfoPG objectives in continuous domains, such as the introduced Multiwalker. Note that, in practice and for simplicity, the policy network can only output the mean value, μ, while the STD value is fixed to a reasonable constant. Finally, to compute the MAP action for a continuous action-space (Line 12 in algorithm 6) we note that, theoretically, the MAP action for a continuous space is just the mean action without any standard deviation from the normal distribution.

D.3 InfoPG Objective Function Derivation (Equation 7.2)

The InfoPG definition as presented in Equation 7.2 consists of a summation across all agents within the communication range. This equation is a simplification from the original form which uses the assumption of independence between each agents' $(k-1)$-level action probability distributions. To arrive at the InfoPG objective function in Equation 7.2, we start from Equation D.2 and convert the log probability of the conditional distributions across all neighbors to a summation of log probability across all neighbors. The process can be shown as in Equation D.2-Equation D.4.

$$\nabla_{\theta}^{\text{InfoPG}} J(\theta) = \mathbb{E}_{\pi^i_{tot}} \left[G^i_t(o^i_t, a^i_t) \nabla_\theta \log(\pi^i_{tot}(a^{i,(k)}_t | \forall_{0\to k} \forall_{j\in\Delta^i_t} [a^{i,(k-1)}_t, a^{j,(k-1)}_t], o^i_t)) \right] \tag{D.2}$$

$$= \mathbb{E}_{\pi^i_{tot}} \left[G^i_t(o^i_t, a^i_t) \nabla_\theta \log(\prod_{0\to k} \prod_{j\in\Delta^i_t} \pi^i_{tot}(a^{i,(k)}_t | a^{i,(k-1)}_t, a^{j,(k-1)}_t, o^i_t)) \right] \tag{D.3}$$

$$= \mathbb{E}_{\pi^i_{tot}} \left[G^i_t(o^i_t, a^i_t) \sum_{j\in\Delta^i_t} \nabla_\theta \log(\pi^i_{tot}(a^{i,(k)}_t | a^{i,(k-1)}_t, a^{j,(k-1)}_t, \dots, a^{i,(0)}_t, a^{j,(0)}_t, o^i_t)) \right] \tag{D.4}$$

D.4 Convergence Proof Sketch for Equation 7.2

Following the approach in prior work [323, 63], we present a convergence proof sketch for InfoPG through the Two-Time-Scale (TTS) stochastic approximation method, proposed by [324]. We note that the convergence proof for InfoPG closely follows the general Policy Gradients (PG) convergence approach presented in [323] and [63], and we therefore only focus on presenting the core idea underlying convergence of InfoPG with k-level rationalizability. Herein, we state that since InfoPG and Consensus Update (CU) share the graph-based local communication and the *fully*-decentralized actor-critic training paradigm, all assumptions made by [63] also apply to our work and therefore, we directly adopt the set of assumptions (i.e., specifically, assumptions 4.1 – 4.4) presented in [63] without restating them.

For an actor-critic algorithm, e.g. InfoPG, showing convergence of both the actor and critic simultaneously is challenging because the noisy performance of the actor can affect the gradient on the critic and vice versa. As such, we leverage the TTS approach, which states that in PG methods, the *actor* learns at a slower rate than the *critic* [324]. Therefore, according to TTS, we can show the convergence of InfoPG in two steps: (1) first, we fix the policy and analyze the convergence of the critic and, (2) with a converged critic, we analyze the convergence of the actor.

Step 1: Critic Convergence – We use bar notation in the following to denote vectorized quantities across agents in the environment such that, $\bar{\pi}_\theta = [\pi_\theta^1, \cdots, \pi_\theta^N]$, where N is the number of agents. Moreover, a level-k policy in InfoPG includes two parts: a state-conditional policy at level $k = 0$ and an action-conditional policy for higher levels of k. InfoPG first applies the level zero state-conditional policy to get the initial "guess" action and then recursively applies the action-conditional policy k times to recursively improve the level-k action-decision. We can show this process as $\pi_\theta^i = \pi_\theta^{i,(k \geq 0)}\left(\pi_\theta^{i,(k=0)}(s_t^i)\right)$. PG seeks to maximize the objective function $\bar{J}(\theta)$, shown in Equation D.5, where $\bar{d}_\pi(\bar{s}_t)$ denotes the stationary distribution of the MDP and $\bar{R}$ denotes the joint reward function, including local rewards for each agent. Additionally, we denote the transition probability of the MDP as $\bar{p}(\bar{s}_{t+1}, \bar{r}_{t+1} | \bar{s}_t, \bar{a}_t)$. Since the formulation of PG objective in Equation D.5 is biased, it is commonly replaced with the unbiased estimate of the rewards, or $\bar{Q}(\bar{s}_t, \bar{a}_t) - \bar{V}(\bar{s}_t, \bar{a}_t)$, as shown in Equation D.6

$$\bar{J}(\theta) = \sum_{\bar{s}_t \in \bar{\mathscr{S}}} \bar{d}_\pi(s) \sum_{\bar{a}_t \in \bar{\mathscr{A}}} \bar{\pi}_\theta(\bar{a}_t|\bar{s}_t) * \bar{R}(\bar{s}_t, \bar{a}_t) \tag{D.5}$$

$$= \sum_{\bar{s}_t \in \bar{\mathscr{S}}} \bar{d}_\pi(s) \sum_{\bar{a}_t \in \bar{\mathscr{A}}} \bar{\pi}_\theta(\bar{a}_t|\bar{s}_t) * (\bar{Q}(\bar{s}_t, \bar{a}_t) - \bar{V}(\bar{s}_t)) \tag{D.6}$$

The unbiased estimator in Equation D.6 can be replaced with the state-value function by using the recursive definition of the action-value function [126]. This substitution results in a form known as the TD-error [64], where the bracketed term in Equation D.8 is the TD-error.

$$\bar{Q}(\bar{s}_t, \bar{a}_t) = \mathop{\mathbb{E}}_{\bar{s}_t \sim \bar{d}_\pi \bar{a}_t \sim \bar{\pi}_\theta} \left[\bar{R}(\bar{s}_t, \bar{a}_t) + \gamma \bar{V}(\bar{s}_{t+1})\right] \tag{D.7}$$

$$\bar{J}(\theta) = \sum_{\bar{s}_t \in \bar{\mathscr{S}}} \bar{d}_\pi(s) \sum_{\bar{a}_t \in \bar{\mathscr{A}}} \bar{\pi}_\theta(\bar{a}_t|\bar{s}_t) * \left[\bar{R}(\bar{s}_t, \bar{a}_t) + \gamma \sum_{\bar{s}_{t+1} \in \bar{\mathscr{S}}} \bar{p}(\bar{s}_{t+1}|\bar{s}_t, \bar{a}_t) \bar{V}(\bar{s}_{t+1}) - \bar{V}(\bar{s}_t)\right] \tag{D.8}$$

Next, following the prior work [323, 63], we assume linear function approximation since the TD-learning-based policy evaluation may fail to converge with nonlinear function approximation [325, 63]. We note that the value function is a mapping of states (of some dimensionality) to $\mathscr{R}$. Therefore, we define $\bar{V}(\bar{s}_t) = \bar{\omega}^T \phi(\bar{s}_t)$, where $\bar{\omega}$ is a one dimensional weight vector and $\phi(\bar{s}_t)$ is a feature map that transforms the state vector to $\mathscr{R}^K$: $\phi(\bar{s}_t) = [\bar{\phi}_1(\bar{s}_t), \cdots, \bar{\phi}_K(\bar{s}_t)]$. Now, following [323], we define the Ordinary Differential Equation (ODE) associated with the recursive update of $\bar{\omega}$ via Equation D.9, which then can be simplified to Equation D.10 using a matrix notation described in the following.

$$\dot{\bar{\omega}} = \sum_{\bar{s}_t \in \bar{\mathscr{S}}} \bar{d}_\pi(s) \sum_{\bar{a}_t \in \bar{\mathscr{A}}} \bar{\pi}_\theta(\bar{a}_t|\bar{s}_t) * \left[\bar{R}(\bar{s}_t, \bar{a}_t) + \gamma \sum_{\bar{s}_{t+1} \in \bar{\mathscr{S}}} \bar{p}(\bar{s}_{t+1}|\bar{s}_t, \bar{a}_t) \bar{\omega}^T \phi(\bar{s}_t + 1) - \bar{\omega}^T \phi(\bar{s}_t) \right] \tag{D.9}$$

$$\dot{\bar{\omega}} = \Phi^T D[T(\Phi\bar{\omega}) - \Phi(\bar{\omega})] \tag{D.10}$$

Finding the asymptotic equilibrium of the critic, ω is equivalent to solving the above equation, when $\dot{\bar{\omega}} = 0$, which is simplified when switching to matrix vector notation described below:

1. $D \in \mathbb{R}^{|\bar{S}| \times |\bar{S}|}$ is a diagonal matrix with $\bar{d}_\pi(\bar{s}_t)$ for all $\bar{s}_t \in \bar{S}$ as its elements.

2. $P \in \mathbb{R}^{|\bar{S}| \times |\bar{S}||\bar{A}|}$ is the probability matrix where $\bar{p}(\bar{s}_{t+1}|\bar{s}_t, \bar{a}_t)\bar{\pi}_\theta(\bar{s}_t|\bar{a}_t)$ represents an individual element.

3. $\Phi \in \mathbb{R}^{|\bar{S}| \times \bar{K}}$ is the feature map whose rows are $\phi(\bar{s}_t)$ for all $\bar{s}_t \in \bar{S}$.

4. $R \in \mathbb{R}^{|\bar{S}| \times |\bar{A}|}$ is a matrix where $\bar{R}(\bar{s}_t, \bar{a}_t)$ represents an individual element.

5. $\Omega \in \mathbb{R}^{|\bar{K}| \times |\bar{K}|}$ is a diagonal matrix with discount factor γ as its elements.

6. $T : \mathbb{R}^N \to \mathbb{R}^N$ is an operator which is a mapping of the form: $T(\bar{\omega}) = R + P\Omega\bar{\omega}$.

Next, following [63], we make two assumptions that apply to InfoPG and are essential to the rest of the convergence proof presented in the following.

Assumption 1 – The update of the policy parameter θ includes a local projection operator, $\Gamma : \mathbb{R}^N \to \chi \subset \mathbb{R}^N$, that projects any θ onto the compact set χ. Also, we assume that χ is large enough to include at least one local minimum of $\bar{J}(\theta)$.

Assumption 2 – The instantaneous reward r_t^i is uniformly bounded for any $i \in \mathscr{N}$ and $t \geq 0$. We note that the reward boundedness assumption is rather mild and is in accordance with prior work [63].

In analogous matrix-vector notation, and under the aforementioned assumptions made above and by [63], for a static policy in the TTS convergence setting, the $\lim_{t\to\infty}\bar{\omega} = \bar{\omega}^{\star}$ almost surely, where $\bar{\omega}^{\star}$ satisfies the equilibrium constraint $\dot{\bar{\omega}} = 0$ shown below. We note the solution seen below satisfies a similar convergence equation as seen in [323].

$$\dot{\bar{\omega}} = \Phi^T D[T(\Phi\bar{\omega}^{\star}) - \Phi(\bar{\omega}^{\star})] = 0 \tag{D.11}$$

$$= \Phi^T D T(\Phi\bar{\omega}^{\star}) = \Phi^T D\Phi(\bar{\omega}^{\star}) \tag{D.12}$$

The above ODE explains the rate of change of the critic, and when the derivative reaches zero, the critic has reached a stable equilibrium, and therefore, has converged.

Step 2: Actor Convergence – According to the TTS [324], for the actor step, we assume a fixed, converged critic and show a stabilized equilibrium of the policy. We assume there exists an operator, Γ, which projects any vector $x \in \mathscr{R}^N \to \chi \subset \mathscr{R}^N$, where χ represents a compact set bounded by a simplex in $\mathscr{R}^N$. The use of this projection is critical to the convergence of stochastic TTS algorithms as stated by [323, 63], since policies that exist outside of the set can cause unstable equilibrium. Empirically, we apply the compactness of the set of policy gradients by defining a maximum gradient norm, as stated in section D.8. In Equation D.13, we define the Γ operator for the vector field $x(.) \in \theta$, which is assumed to be a continuous function.

$$\hat{\Gamma}(x(y)) = \lim_{0\leq\eta\to 0} \frac{\Gamma(y+\eta x(y)) - y}{\eta} \tag{D.13}$$

If the above limit does not converge to a singular value, we state $\hat{\Gamma}(x(y))$ results in a *set* of convergent points. With this notation we state the ODE of the policy, after being projected onto a compact set, and note that given the assumptions made above and by [63], PG almost surely moves $\bar{\theta}$ to an asymptotically stable equilibrium that satisfies the below Equation D.14. This proof analogy closely follows the single-agent convergence proofs presented in [323] and [325]. Nevertheless, in our work, convergence to a stationary point for all agents is the goal. While the TTS approach assumes the critic to have converged, the critic does not need to be a perfect estimator. With small approximation error, [323] proves that the below equation still converges within the neighborhood of the optimal concatenated, joint policies for all agents. The below ODE defines the derivative of the policy over time, and as the derivative approaches zero, the actor reaches a stable equilibrium (or a set of equilibrium points, since in a fully decentralized setting, the actor is a vector of joint policies for all agent) and thus, convergence of the actor is achieved.

$$\dot{\bar{\theta}} = \mathbb{E}_{\bar{s}_t \sim \bar{d}_\pi, \bar{a}_t \sim \bar{\pi}_\theta} \left[\nabla \log(\bar{\pi}_\theta) * (\bar{Q}(\bar{s}_t, \bar{a}_t) - \bar{V}(\bar{s}_t)) \right] = 0 \tag{D.14}$$

D.5 Full Proof of Theorem Theorem 5

Here, we derive the full proof of the Bayesian expansion of the policy (Theorem Theorem 5). Without loss of generality, we assume a scenario with two cooperating agents i and j both with k levels of rationalization. An important notational difference that will be used here is $p(.)$. Policies are conventionally considered state-conditional distributions of actions, where the action is the random variable. A *specific* action is usually either sampled from the distribution or the MAP action is selected. Here, we denote the probability distribution of a particular random variable X as $p(X)$. Note that, evaluating $p(X = x)$ returns the specific probability that $X = x$ (and not a distribution). We first define the joint density, $p(a_t^{i,(k)}, a_t^{j,(k)})$ by marginalizing $p(a_t^{i,(k)}, a_t^{j,(k)}, a_t^{i,(k-1)}, a_t^{j,(k-1)})$:

$$p(a_t^{i,(k)}, a_t^{j,(k)}) = \sum_{x \in \mathscr{A}} \sum_{y \in \mathscr{A}} p(a_t^{i,(k)}, a_t^{j,(k)}, a_t^{i,(k-1)} = x, a_t^{j,(k-1)} = y) \tag{D.15}$$

The interior marginal can be further broken down into each agents' conditional k-level conditional policy through the the Bayesian formulation of the k-level decision hierarchy:

$$p(a_t^{i,(k)}, a_t^{j,(k)}) = \sum_{x\in\mathscr{A}}\sum_{y\in\mathscr{A}} p(a_t^{i,(k)}|a_t^{i,(k-1)} = x, a_t^{j,(k-1)} = y)p(\\ a_t^{j,(k)}|a_t^{i,(k-1)} = x, a_t^{j,(k-1)} = y)p(a_t^{i,(k-1)} = x, a_t^{j,(k-1)} = y) \tag{D.16}$$

The joint density, $p(a_t^{i,(k)}, a_t^{j,(k)})$ can also be formulated using its conditional definition, which we combine with the uniform prior assumption $(p(a_t^{j,k}) = \frac{1}{\mathscr{A}})$:

$$p(a_t^{i,(k)}, a_t^{j,(k)}) = p(a_t^{i,(k)}|a_t^{j,(k)})p(a_t^{j,(k)}) \tag{D.17}$$

$$p(a_t^{i,(k)}|a_t^{j,(k)}) = |\mathscr{A}|p(a_t^{i,(k)}, a_t^{j,(k)}) \tag{D.18}$$

From here, we can substitute the joint density marginal defined in Equation D.5 into the conditional density defined in Equation D.17. By considering $|A|$ and the joint density within the marginal in Equation D.5 as weightage factors, we can draw the following proportionality relation for all $a_t^{i,(k-1)} = x, a_t^{j,(k-1)} = y \in \mathscr{A}^2$:

$$p(a_t^{i,(k)}|a_t^{j,(k)}) \propto p(a_t^{i,(k)}|a_t^{i,(k-1)}, a_t^{j,(k-1)})p(a_t^{j,(k)}|a_t^{i,(k-1)}, a_t^{j,(k-1)}) \tag{D.19}$$

In Eq. Equation D.19, the conditional densities, whom are conditioned on the $k-1$ actions of i and j, are exactly the same as the k-level communicative policy definition. These substitutions are implemented in Equation D.20.

$$\pi^i_{\text{com}}(a_t^{i,(k)}|a_t^{j,(k)}) \propto \pi^i_{\text{com}}(a_t^{i,(k)}|a_t^{i,(k-1)}, a_t^{j,(k-1)})\pi^j_{\text{com}}(a_t^{j,(k)}|a_t^{i,(k-1)}, a_t^{j,(k-1)}) \tag{D.20}$$

During InfoPG, we seek to maximize the total policy, π^i_{tot}, which is achieved by performing gradient ascent. The total policy is a direct concatenation of the communicative policy, π^i_{com} and the encoding policy, π^i_{enc}, so if we maximize the total policy, then we also maximize the individual communicative and encoding policies. Therefore, since we individually maximize both terms in the RHS of Equation D.20, we can assert that we are also proportionally increasing $\pi^i_{\text{com}}(a_t^{i,(k)}|a_t^{j,(k)})$, as seen in Equation D.21, which is the conclusion drawn at the end of subsubsection 7.1.4.

$$\nabla\pi^i_{\text{com}}(a_t^{i,(k)}|a_t^{i,(k-1)},a_t^{j,(k-1)})\nabla\pi^j_{\text{com}}(a_t^{j,(k)}|a_t^{i,(k-1)},a_t^{j,(k-1)}) \rightarrow\propto \nabla\pi^i_{\text{com}}(a_t^{i,(k)}|a_t^{j,(k)}) \tag{D.21}$$

D.6 Discussion on the Uniformity of Priors Assumption

Similar assumption of uniform priors as ours in **??** have been used previously by [277] for the calculation of MI upper-bound. In our work, we defined $p(a_i)$ as a marginalization of the action-conditional policy, $\pi^i_{com}(a^i|a^j)$, across any potential a^j. The marginal's conceptual meaning here is similar to asking the question, "*What should the probability of a^i be, if we did not know a^j?*" For a given action-conditional policy, we could expect a^i to be uniformly random because the action-conditional policy is expected to only change the probability of a selected action based on the k-level reasoning with other agents. If there is no action to reason upon, the agent has no information with which to base its k-th action upon. It is important to note here that $p(a^i)$ is not a marginalization of both the state-conditional and action-conditional policies. Since $p(a^i)$ is only a marginalization of the action-conditional policy, we view the uniformly-random prior assumption as a reasonable design choice.

D.7 Supplementary Results

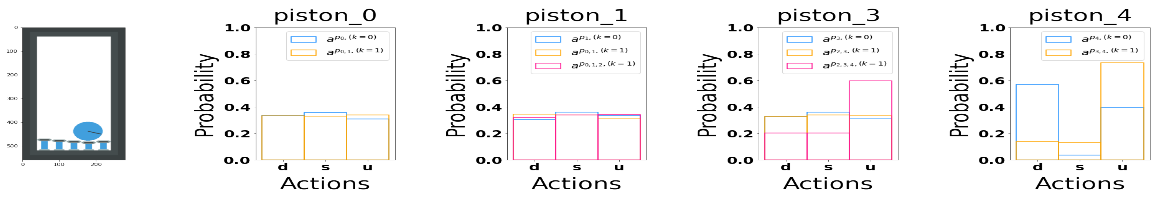
In this section we provide our supplementary results. We start by analysing and interpreting agents' communicative policy in our fraudulent agent experiment in order to understand the effect of k-level rationalization for decision-making in this scenario. Next, we present the agent-level performances for InfoPG and compare with the baselines in all three evaluation environments. Next, we present an scalability analysis in the Pistonball environment to investigate robustness to larger number of agents. Eventually, we conclude this section by presenting a list of takeaways.

D.7.1 Policy Interpretation for the Fraudulent Agent Experiment

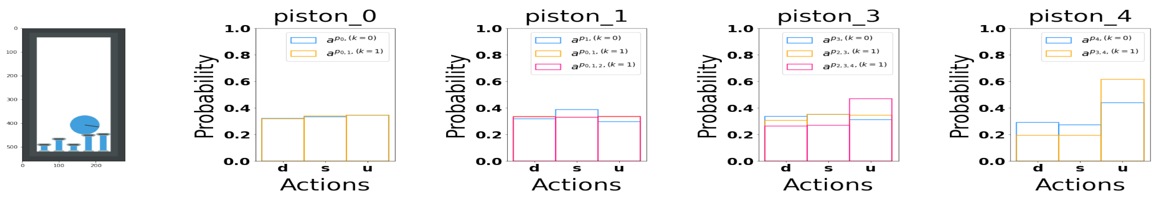
In this section, we present an analysis and interpretation of agents' communicative policy in our fraudulent agent experiment in order to understand the effect of *k*-level rationalization for decision-making in this scenario. We test the learnt policy at convergence using Adv. InfoPG in the fraudulent agent experiment, presented in **??**. The results are shown in Figure D.1 in which, we present the action distributions of agents before and after rationalizing their decisions with their neighbors through *k*-level reasoning. For each action distribution graph, the Y-axis is the probability of an action and the X-axis represents actions, where *u*=*move up*, *d*=*move down*, and *s*=*stay* constant.

In Figure Figure D.1 we present sample illustrations of a test run in a BGP scenario for $t = 0, 10, 25, 32, 37$, from start to the end of the episode. At $t = 0$ (Figure D.1a), we can see that the episode starts with the ball on top of the right-most pistons (i.e. pistons 3 and 4). Note that pistons are indexed left to right. With $k = 1$ reasoning, we show piston 3's rationalization for its action decision in three phases: first, $a^{p_3,(k=0)}$, is the naive $k = 0$ action (blue); second, $a^{p_{2,3},(k=1)}$, is the $k = 1$ rationalization that incorporates piston 2's random action (orange), and, third, $a^{p_{2,3,4},(k=1)}$, is the $k = 1$ rationalization that incorporates both piston 2's random action and piston 4's $k = 0$ action into piston 3's $k = 1$ action decision (pink). Using a similar notation convention, $a^{p_{3,4},(k=1)}$ is piston 4's action rationalization given piston 3's $k = 0$ action rationalization. Now, notice that the spread of $a^{p_3,(k=0)}$ at $t = 0$ is relatively uniform (blue), and given that piston 2 is randomly moving, $a^{p_{2,3},(k=1)}$ remains unchanged (orange). This indicates piston 3 has learned to *ignore* piston 2, which is the fraudulent agent. Additionally, notice that $a^{p_4,(k=0)}$ at $t = 0$ is bimodal and has relatively equal probabilities of moving either *up* or *down*; however, we can see for both pistons 3 and 4, after rationalizing their actions with each other at $k = 1$, both action distributions become

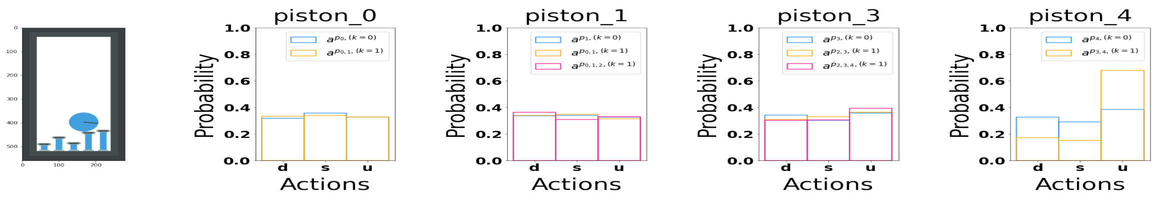
(a) Action distributions for pistons at $t = 0$



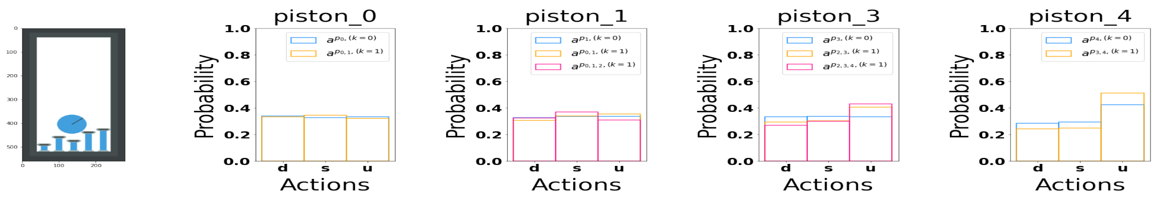
(b) Action distributions for pistons at $t = 10$



(c) Action distributions for pistons at $t = 25$



(d) Action distributions for pistons at $t = 32$



(e) Action distributions for pistons at $t = 37$

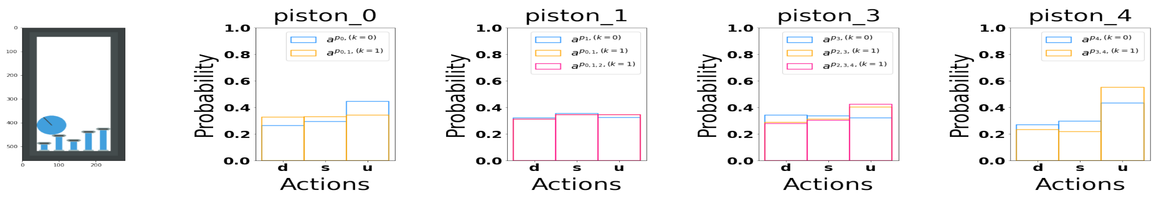


Figure D.1: Action distributions of piston agents across 37 timesteps for the fraudulent agent experiment introduced in section 7.1. Note that Piston 2 (not displayed) is the fraudulent agent with untrainable random policy.

unimodal and tend towards moving up, which is the desired action for moving the ball to left.

This coordination demonstrates an interesting strategy; piston 3 and piston 4 have learned that coordinating actions with the randomly moving piston 2 is not desirable and therefore, they seek to move the ball as high as possible, and toss it over piston 2. Empirical proof of this behavior can be seen by the continuation of the spread of distributions at $t = 10$. At $t = 25$ a distinct change occurs; piston 3's action distribution, after $k = 1$ rationalization with piston 4, becomes uniform again, while Piston 4 is still unimodal and tending up. We believe this behavior shows that piston 3 and 4 have realized the strategy to bypass piston 2 is to "launch" the ball over piston 2, which can be accomplished by piston 4 moving up, while piston 3 remains stable, effectively creating a leftward force for the ball to move left. At $t = 32$ we can see the "launching" is performed, and here the action distributions of both piston 3 and 4 become relatively uniform again (Note that actions of piston 3 and 4 do not matter at this point since they are not directly located under the ball and therefore, do not receive a reward for their actions). From $t = 32$ to $t = 37$, the ball traverses over pistons 0 and 1; however, note that piston 0 and 1 will not need to move the ball too much, since the ball already has leftward momentum. Accordingly, piston 0 and 1 only need to coordinate to create a leftward ramp to facilitate ball's movement. As such, both piston 0 and piston 1 follow relatively uniform distributions after k=0 and 1 rationalization. At $t = 37$, the ball is over piston 0 and has reached the left wall, which denotes winning and end of the episode.

In summary, we show two key behaviors learnt by agents through our Adv. InfoPG in the fraudulent agent experiment: 1) Piston 2 is untrustworthy and thus, coordinating with this agent is not desired, which leads to unchanged action-distributions for Pistons 3 and 1 after iterated k-level reasoning with this fraudulent agent. 2) Pistons 3 and 4 learn to avoid the fraudulent agent (piston 2) by "launching" the ball over it, giving the ball a leftward momentum to reach the left wall.

D.7.2 A Qualitative Analysis for Bounded Rationality

The postulate of k-level reasoning is that higher levels of k should allow for deeper decision rationalization and therefore better strategies. In this section, we qualitatively investigate *different examples of intelligent behavior induced by varying bounds of rationality*. To address this, we specifically further investigate InfoPG's results in the MultiWalker and StarCraft II (SC2) domains due to their complex mechanics and multi-faceted objectives. In the following, we first present our qualitative analysis for SC2 and Multiwalker, respectively.

SC2 – Our qualitative analysis in SC2 is a demonstration of how bounded rationality and higher levels of iterated reasoning benefits performance. In SC2, agents are positively rewarded for shooting and killing enemy agents, and are negatively penalized for getting shot at. Therefore, a locally optimal strategy is to run away from the enemy team to avoid any negative penalties, while a globally optimal strategy is to kill and eliminate all the enemy agents to achieve high positive rewards.

Figure D.2 shows our qualitative results for analyzing the effects of assuming bounded-rational agents and iterated reasoning in the SC2 domain. We compare the learned policies by InfoPG at convergence in the SC2 domain with $k = 0$ and $k = 1$.

At $k = 0$ of the rationalization hierarchy, the fully naive and non-strategic level-0 agents choose actions while completely disregarding other agents actions (i.e., have zero-order beliefs). As such, for a level-0 policy, we expect to observe that agents simply run away from the enemy to avoid getting shot at, since a single agent does not believe (zero-order belief) it can overcome the enemy team. As seen in Fig. 5.1–5.3, the naive agents expectedly only learn to run away to avoid the negative penalties of getting shot at. This fleeing behavior allows agents to maintain a reward of zero, as shown in Figure 7.2, indicating successful escape and convergence to the locally optimum solution.

At level $k = 1$, each agent is now more sophisticated and believes that the other agents have a level-0 policy and takes actions according to that. In this case, we observe a vastly different behavior. As shown in Fig. 5.4–5.6, agents here learn more strategic policies to

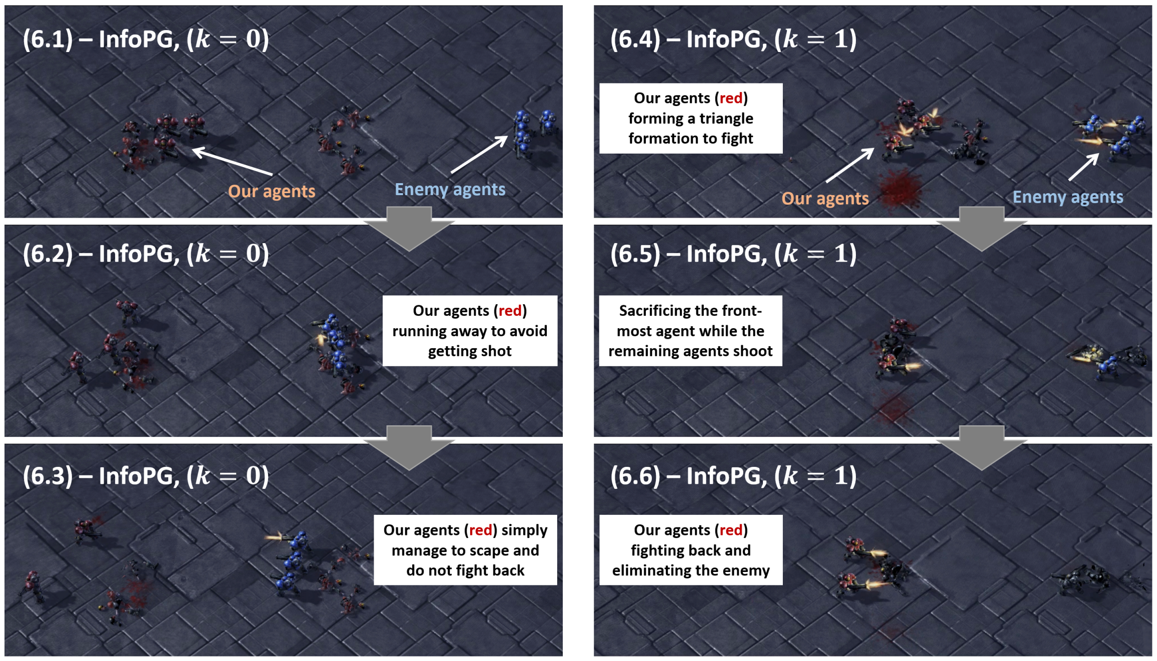

(6.1) – InfoPG, ($k = 0$)
Our agents
Enemy agents
(6.2) – InfoPG, ($k = 0$)
Our agents (red) running away to avoid getting shot
(6.3) – InfoPG, ($k = 0$)
Our agents (red) simply manage to scape and do not fight back
(6.4) – InfoPG, ($k = 1$)
Our agents (red) forming a triangle formation to fight
Our agents
Enemy agents
(6.5) – InfoPG, ($k = 1$)
Sacrificing the front-most agent while the remaining agents shoot
(6.6) – InfoPG, ($k = 1$)
Our agents (red) fighting back and eliminating the enemy

Figure D.2: Comparing the learned policies by InfoPG at convergence in the SC2 domain with $k = 0$ and $k = 1$. With $k = 0$ (Fig. 5.1–5.3), the naive agents disregard other agents actions and simply learn to run away to avoid the negative penalties of getting shot at. This is while the more sophisticated agents with $k = 1$ (Fig. 5.4–5.6), learn more strategic policies to work together to eliminate the enemy team and achieve high positive rewards.

work together to eliminate the enemy team and achieve high positive rewards. The agents begin a triangle-like formation towards the enemy (5.4). Enemy agents then begin to shoot the closest opposing player at the front of the triangle formation. The other two agents in the team use this opportunity and start firing at the enemy team while they shoot the front-most agent. As such, the remaining two agents manage to kill and eliminate the enemy team. Through reasoning their level-1 actions based on their teammates level-0 action, InfoPG agents learn a sacrificial technique of exposing one agent as bait, which allows the agents to converge to the globally optimum solution of killing the entire enemy team. This is also reflected in Figure Figure 7.2, where the $k = 1$ InfoPG achieves the highest cumulative rewards.

Multiwalker – Our qualitative analysis in Multiwalker is another demonstration of how higher levels of iterated reasoning benefits performance. There are two objectives that the walkers need to satisfy: (1) stabilization (both the package and the walkers) and (2)

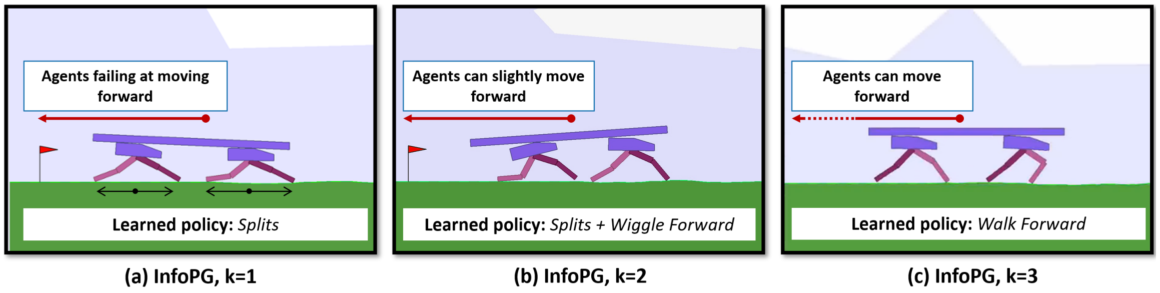

Agents failing at moving forward
Learned policy: *Splits*
(a) InfoPG, k=1
Agents can slightly move forward
Learned policy: *Splits + Wiggle Forward*
(b) InfoPG, k=2
Agents can move forward
Learned policy: *Walk Forward*
(c) InfoPG, k=3

Figure D.3: Comparing the learned policies by InfoPG in the Multiwalker with $k = 1$, $k = 2$, and $k = 3$, 5a-5c, respectively. With $k = 1$, agents only learn to perform a split to balance the package on top and avoid falling. This is while with $k = 3$ agents learn to quickly walk forward. The middle stage of rationalization, $k = 2$, achieves an in-between policy where agents split to balance, but also wiggle forward slowly.

moving forward. Stabilization of the package and the walkers are the primary goals, since dropping the package, or falling, results in failing the game with a penalty. Walking (i.e., moving forward to the right) is an additional goal, since every forward step yields a small proportional reward. Ultimately, the walkers should aim to achieve both stabilization and walking at the same time, which is the globally optimum solution.

Figure D.3 shows our qualitative results for analyzing the effects of assuming bounded-rational agents and iterated reasoning in the Multiwalker domain. At level $k = 1$, each walker believes that the other walker has only a level-0, non-rational policy. We observe in Figure D.3-(a) that, with InfoPG and by $k = 1$, the walkers solve only the stabilization problem by learning to do the ”splits“, which is a locally optimum solution. The walkers create a wide base with their legs and simply hold the package statically with no forward progress (evidenced by the starting red flag). This technique requires some degree of coordination since the walkers have to do the splits synchronously at the beginning of the experiment; however, this is not nearly a complex enough strategy to achieve any positive rewards. Looking at the reward convergence in Figure 7.2, $k = 1$ achieves converges to the locally optimum solution and achieves a reward of 0, since the walkers do not get any positive reward for moving nor do they get any negative reward for dropping the package or falling.

At level $k = 2$, each walker now believes that the other walker has a level-1, bounded-rational policy. Intuitively, assuming a more sophisticated policy for a teammate should lead to a better overall strategy, since the best-response solution to such sophisticated policy needs a certain level of sophistication [326, 269]. We observe in Figure D.3-(b) that, as expected, the learned policies at level $k = 2$ of rationalization still includes performing the ”splits“ for balancing while agents also learned to wiggle forward slowly and receive some positive reward.

As we increase the rationalization depth from $k = 3$, we see in Figure D.3-(c) that the walkers not only stabilize the package, but also start moving forward (evidenced by the starting red flag out of frame) with much more sophisticated strategies. The left-most walker learns to generate forward momentum and walk more quickly than the front walker, which learns to walk more slowly and maintain the stability of the package. Note that this illustrates the idea of role allocation, which is a relatively complex strategy and indicative of higher levels of intelligence achieved through assuming sophisticated, level-2 teammates. The walkers learn to coordinate their movements, because if the left-most walker makes too jerky of a forward movement, the right-most walker adjusts by staying more static to stabilize the package. In Figure 7.2, the collective strategy at $k = 3$ can achieve rewards as high as +10, which is the globally optimum solution.

D.7.3 Scalability Analysis: Pistonball

Here, we investigate InfoPG's robustness to larger number of interacting agents in the Pistonball environment. For this experiment, We selected our best-performing model, Adv. InfoPG, and the best-performing baseline, MOA [99], from our primary results in **??**. We increased the number of agents from five to ten and kept the communication range to be the same (i.e., one piston on each side). The results are presented in Figure D.4. As shown, Adv. InfoPG outperforms MOA in both maximizing average individual and team reward performances during training.

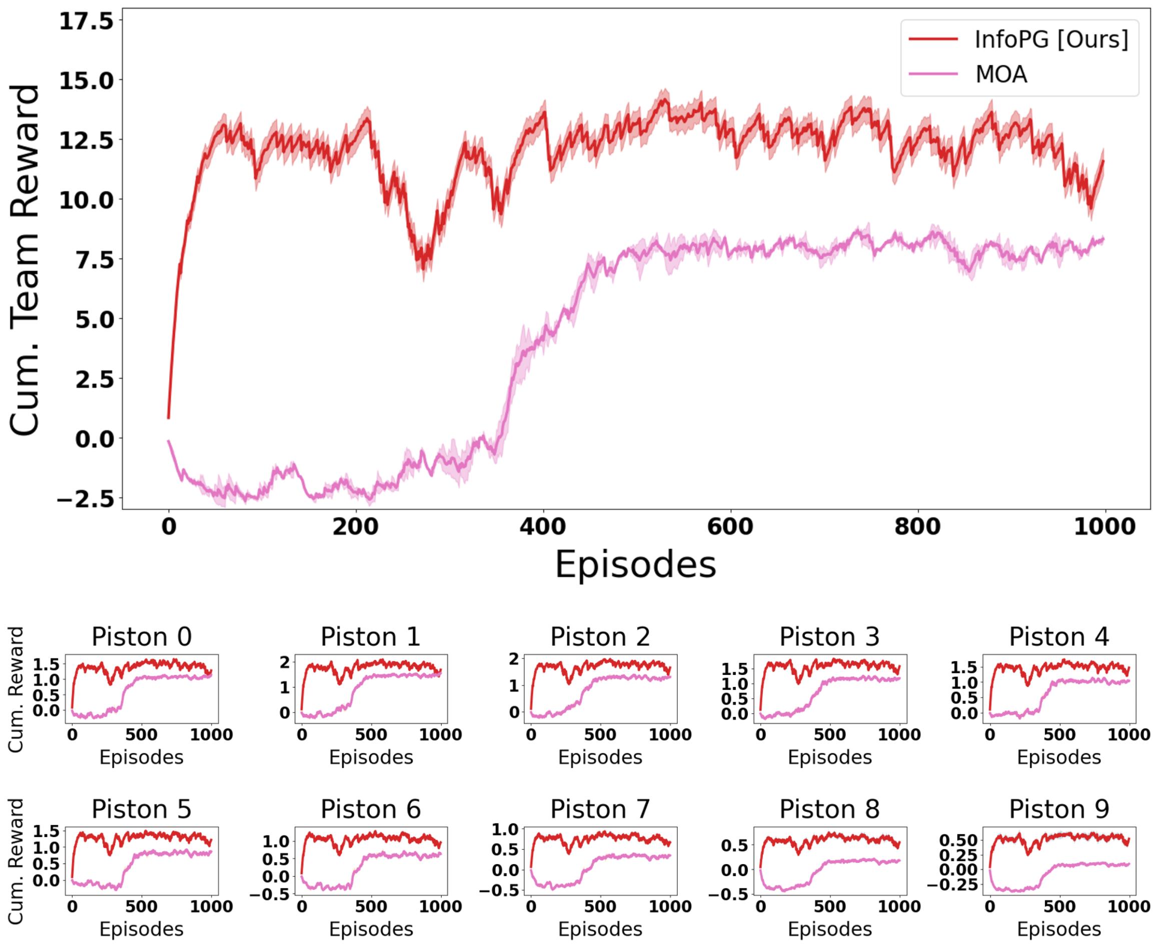


Figure D.4: Scalability comparison between Adv. InfoPG and the best-performing baseline, MOA [99], in the Pistonball domain with ten interacting agents. Adv. InfoPG outperforms MOA in both maximizing average individual and team reward performances.

InfoPG considers two way communication with each of its neighbors (there are $|\Delta|$ neighbors which are communicated with k times). If $\mathscr{D}$ is the dimension of the communicated k-level action-vector, the bandwidth of input and output communication channels is $\Theta(2|\Delta|k\mathscr{D})$, where each communication channel is $\Theta(|\Delta|k\mathscr{D})$. We leave the choice of $|\Delta|$, k, and $\mathscr{D}$ to be hyper-parameters, all of which can be lessened as the number of agents increase to inhibit computational complexity issues.

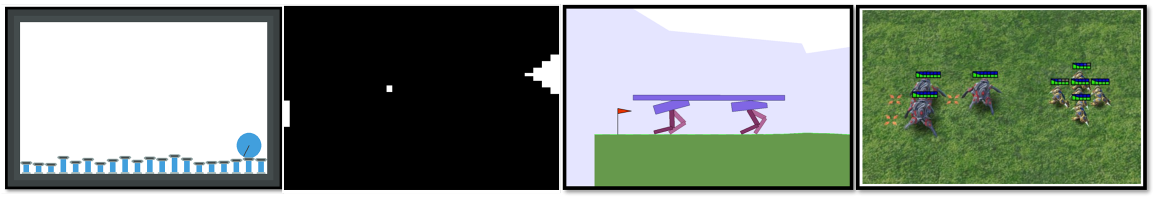


Figure D.5: Instances of the utilized multi-agent cooperative environments. Domains are parts of the PettingZoo [282] MARL research library and can be accessed online at `https://www.pettingzoo.ml/envs`. The StarCraft II [284], can be accessed from Deepmind's repository available online at `https://github.com/deepmind/pysc2`.

D.7.4 Agent-wise Performance Comparison

As mentioned, the objective in a *fully* decentralized domain is to maximize the average return by each individual agent, such that the obtained cumulative team reward is also maximized. We have show in our primary results in **??** that InfoPG outperforms all baselines across all three domains in achieving higher cumulative team reward. Here, we present the agent-level reward performances for InfoPG and compare with the baselines across three domains. The results are presented in Figure D.6, where sub-Figure D.6a-Figure D.6d represent the individual agent performances in Co-op Pong, Pistonball, Multiwalker and StarCraft II (3M), respectively. As shown, our InfoPG and its MI-regularizing variant, Adv. InfoPG, continually outperform the other baselines in maximizing achieved individual rewards for agents. Specifically, in all graphs for Adv. InfoPG, all agents maximize individual rewards over time, and Adv. InfoPG achieves SOTA across all baselines.

D.8 Evaluation Environments: Additional Details and Parameters

Here we provide additional details regarding the employed evaluation environments for training and testing InfoPG and cover the associated environment parameters related to our experiments. An instance of the four environments are presented in Figure D.5.

1) Cooperative Pong (Co-op Pong) [282] – The objective in this game is to keep a pong ball in play for as long as possible between two co-operating paddles. To fit the MAF-Dec-POMDP paradigm, agents must receive individualistic rewards. In the Co-op Pong domain, each paddle receives a reward of $+1$ if it hits an incoming pong ball successfully and a penalty of -1 if it misses. The game ends either when a paddle misses or if $max\_cycles = 300$ cycles have elapsed. Therefore, in order to continue receiving positive rewards of $+1$, paddles are implicitly encouraged to cooperate to maximize their accumulated rewards. For an episode of the game, the pong ball was set to move at a velocity of $15[\frac{pixels}{sec.}]$ while the paddles move slightly slower at $13[\frac{pixels}{sec.}]$. Since the ball moves faster than the paddles, the paddles require "forecasting" their intended position when the pong ball comes into their field-of-view (FOV), which is a 280×240 RGB image. An important facet of this environment is the time-delayed nature of the actions. Consider a scenario when the left-side paddle hits the ball at t_1; this means that the right-side paddle will receive the ball at minimum, at $t_2 = t_1 + \frac{280}{15}$. This measure is an underestimation, since the pong ball will not likely move in a straight line drive (i.e. it may hit the sides of the walls) but it illustrates the point that the action of the left-side paddle at t_1 is particularly relevant to the action of the right-side paddle at t_2; however, the action of the left-side paddle at t_2 is not particularly relevant to the right-side.

Therefore, in our experiments, to account for the time-delay in the action information, a "hub" was designed where the action at the time of the last "hit" is shared to the opposing paddle, and a zero-vector otherwise. This procedure was applied for both InfoPG and MOA [99] as to fairly evaluate the communicative algorithms. In the case of MOA, the time-delayed action was sought to be "predicted" by the opposing paddle's model of agent.

2) Pistonball [282] - The goal in the this environment is to move a ball from right side of the screen (e.g., right wall) to the left side of the screen (e.g., left wall) by activating/deactivating a team of five vertically moving piston agents. The ball has momentum in motion and is elastic. In order to encourage robustness, the ball was randomly placed on the pistons with a friction factor of 0.3, mass of 0.75, and a relatively high elasticity factor of 1.5. An episode of the game ends if agents move the ball to the left wall or after *max_cycles* $= 200$ cycles have elapsed. Each agent's observation is an RGB image of size 457×120 covering the two pistons (or the wall) around an agent and the space above them. Each piston receives an individual reward which is a combination of how much the corresponding agent *directly* contributed to moving the ball to left (i.e., with a value of $\mathscr{X}_t^{ball} - \mathscr{X}_{t-1}^{ball}$, where $\mathscr{X}_t$ represents the center position of the ball along the X-axis at time t), and a negative time penalty of -0.007 per timestep. A piston is considered to be contributing directly to moving the ball to left, if it is directly below any part of the ball. Given the ball's radius of 20 pixels, at each timestep, three pistons can be directly under the ball. Agents win an episode of the game if they can coordinate efficiently to move the ball to the left wall within allowed maximum steps.

3) Multiwalker [283, 282] - The objective in this continuous-space environment is for a team of two bipedal robots to carry a heavy package cooperatively and walk as far right as possible. The weight of the package depends on its length which is determined by the number of agents. We note that, the two-agent case is the most challenging scenario in Multiwalker. Each robot exerts a variable force on two joints in its two legs, and therefore, the action-space is of dimension four with values in range $(-1, 1)$. The bipedal robots receive local rewards related to individual balance and stability of their hull. The reward function includes a reward of $+1$ for a scaled forward displacement of each bipedal robot's hull. We set the maximum number of allowed steps to *max_cycles* $= 500$ cycles, which would terminate at any point if a bipedal or the package falls. If a bipedal robot falls, it will individually receive a penalty of -10, and, if the package falls on the ground, each bipedal receives a penalty of -100. Each agent receives a 31-dimensional observation vector. The first 24 elements of the observation vector represent the bipedal robot's internal kinematics and the rest (i.e, elements 24-31) relate to LIDAR observations of the package position as well as the position of the adjacent bipedals.

4) StarCraft II [284] (*The 3M (three marines vs. three marines) challenge*) - The goal in this domain is for a team of three friendly marines to find, shoot, and kill three enemy marines as soon as possible, without dying or getting hit [54, 8]. This domains is more challenging than the previous ones since the state-space is larger and the communication graph is time-varying. In this challenge, marines can move in four primitive directions and shoot an enemy marine within distance (i.e., multiple enemy marines can be in vision at once), and therefore, the action-space dimension is also larger than the previous domains. Agents get negatively rewarded when they are shot by enemy marines and are positively rewarded when shooting enemy marines. 3M presents a fundamentally more challenging environment than Pistonball, Multiwalker, and Pong, as the state space is much larger, and agents are allowed to move in 2D over a large gameplay arena. The action space is also larger (size 8) as agents can choose between: no action, moving in 4 directions, and shooting any one out of 3 enemy marines. Additionally, agents have a time-varying communication graph (different from the other domains), because friendly marines move in and out of the line of sight.

D.9 Training and Execution Details

Under the MAF-Dec-POMDP paradigm, each agent $i \in \mathscr{N}$ is equipped with its own optimizer and policy π^i_{tot} which consists of an encoding policy π^i_{enc} and a communicative policy π^i_{com} (each parameterized by θ^i). The encoding policy can be represented using a feed-forward neural network, and the communicative policy can be represented by any class of Recurrent Neural Networks (RNNs), such as the Gated Recurrent Unit (GRU) or Long Short-Term Memory (LSTM). For computational efficiency, we chose to use a GRU or simplistic RNN architecture.

Additionally, while we formulaically denote actions for level k as $a_t^{i,(k-1)}$, in execution we represent these actions as finite-dimensional vectors to maximize information during inference. The size of these vectors are known as *policy latent size* in the provided hyper-parameter tables. This parameter (also shared by other baselines) refers to the size of the latent vector prior to the final Softmax output layer. During the encoding stage of InfoPG, each agent, i, receives an observed state vector, o_i^t, and encodes an action vector $a_t^{i,(0)}$, using π_{enc}^i. During k-level communication, each agent receives the action vectors of neighboring agent $j \in \Delta_t^i$ from level $k-1$ and performs a forward-pass on the RNN, where the initial cell state is $a_t^{i,(k-1)}$. The action probabilities for the discrete domains (i.e., Co-op Pong and Pistonball) are outputted by the feed-forward network where the last layer size is equal to the size of the action space and a Softmax activation. Note that for our continuous action-space domain, Multiwalker, the final Softmax activation function is replaced with the Tanh activation. Neighboring agents are determined using the adjacency graph $\mathscr{G}_t$, and a distance hyper-parameter specifying how "far" agent i can communicate (i.e., communication range). $\mathscr{G}_t$ is an undirected time-varying graph, and as agents perform actions and change their relative position (depending on the domain), the edges $\mathscr{E}_t \subseteq \{(i,j) : i,j \in \mathscr{N}, i \neq j\}$ are updated for the next timestep. This process is carried out until convergence of the cumulative rewards of all N agents.

Specifics for Co-op Pong – The input observation in this domain, a 280×240 RGB image, contains information about where the pong ball exists in the FOV of each paddle. Since the ball is in motion, we found higher performance could be achieved by setting the observation at time, t, to be the difference between the observation at t and $t-1$. As such, we encoded the input observation to represent information about not just the position, but also the velocity of the ball. This procedure was maintained for all baselines.

Another property we found critical to the performance of InfoPG agents in Co-op Pong was the *type* of RNN for π_{com}^i. In Pong, rewards are rather sparse, since paddles only receive feedback when they hit or miss a ball, while in other times and when ball is traversing the screen (which is the majority of the time spent in the game) no feedback is received from the environment. Accordingly, we leveraged curriculum learning [327] such that we let agents first learn the mechanics of hitting the pong ball and then, learn the benefit of communication. We achieved this behavior by using a simple RNN (we distinguish this with VRNN for Vanilla RNN) cell for π^i_{com}, where the initial weight matrix W_{ih} was set to the identity matrix and all other parameters were set to a small constant. This way, we effectively make the output of the $\pi^i_{com} = \pi^i_{enc}$ at the beginning episodes of training, while as time elapses, the weight matrix is optimized to incorporate actions from the neighboring paddle.

Specifics for Pistonball – The agents each receive a 457×120 RGB image as their observation input. In order to minimize feature size, each observation was first cropped to a size of 224×224, normalized and inputted into a pre-trained AlexNet model. AlexNet [328] is a CNN that takes in images and outputs probability scores of classes. In our experiments, we utilized the first four intermediate layers of AlexNet to produce rich feature observations to the input of the encoding policy. This procedure was applied to all baselines.

Hardware Specifics – All experiments were conducted on an NVIDIA Quadro RTX 8000 with approximately 50 GB of Video Memory Capacity.

Training Hyperparameters – We present the training hyperparameters in our implementations and experiments across methods and all three environments in Tables Table D.1-Table D.5.

Table D.1: Co-op Pong Training Hyperparameters.

Experiments	**Pong**					
	InfoPG	**Adv InfoPG**	**NC-A2C**	**CU**	**MOA**	**PR2-AC**
Learning Rate	4e-4	4e-4	4e-4	4e-4	4e-4	4e-4
Size of Latent Vector	30	30	30	30	30	30
Type of Com. Network	VRNN	VRNN	-	-	GRU	-
Epochs	4000	4000	4000	4000	4000	4000
MOA Loss Weight	-	-	-	-	0.1	-
Discount Factor	0.95	0.95	0.95	0.95	0.95	0.99
Batch Size	16	16	16	16	16	16
Max Gradient Norm	10	10	10	10	10	10
Replay Buffer Size	-	-	-	-	-	1e5
Number of Particles	-	-	-	-	-	16

Table D.2: Pistonball Training Hyperparameters.

Experiments	**Pistonball**					
	InfoPG	**Adv. InfoPG**	**NC-A2C**	**CU**	**MOA**	**PR2-AC**
Learning Rate	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
Size of Latent Vector	20	20	20	20	20	20
Type of Com. Network	GRU	GRU	-	-	GRU	-
Epochs	1000	1000	1000	1000	1000	1000
MOA Loss Weight	-	-	-	-	1.0	-
Discount Factor	0.99	0.99	0.99	0.99	0.99	0.99
Batch Size	4	4	4	4	4	4
Max Gradient Norm	0.75	0.75	0.75	0.75	0.75	4.0
Replay Buffer Size	-	-	-	-	-	1e5
Number of Particles	-	-	-	-	-	16

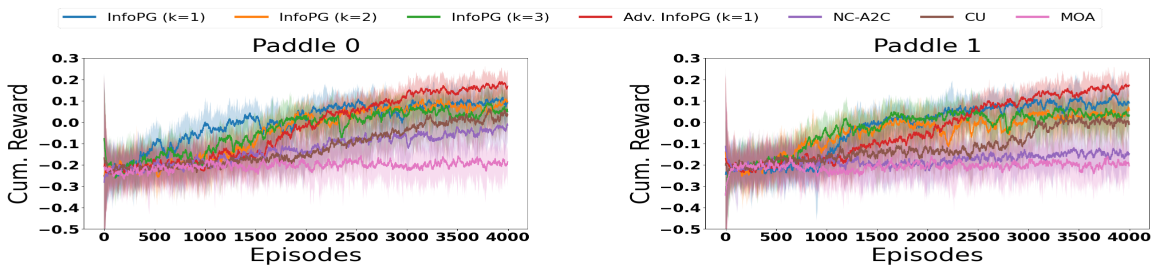
Table D.3: Multiwalker Training Hyperparameters.

Experiments	**Multiwalker**					
	InfoPG	**Adv. InfoPG**	**NC-A2C**	**CU**	**MOA**	**PR2-AC**
Learning Rate	4e-4	4e-4	4e-4	4e-4	4e-4	4e-4
Size of Latent Vector	30	30	30	30	30	30
Type of Com. Network	GRU	GRU	-	-	GRU	-
Epochs	1000	1000	1000	1000	1000	1000
MOA Loss Weight	-	-	-	-	0.1	-
Discount Factor	0.95	0.95	0.95	0.95	0.95	0.95
Batch Size	16	16	16	16	16	16
Max Gradient Norm	5	5	5	5	5	5
Replay Buffer Size	-	-	-	-	-	1e5
Number of Particles	-	-	-	-	-	16

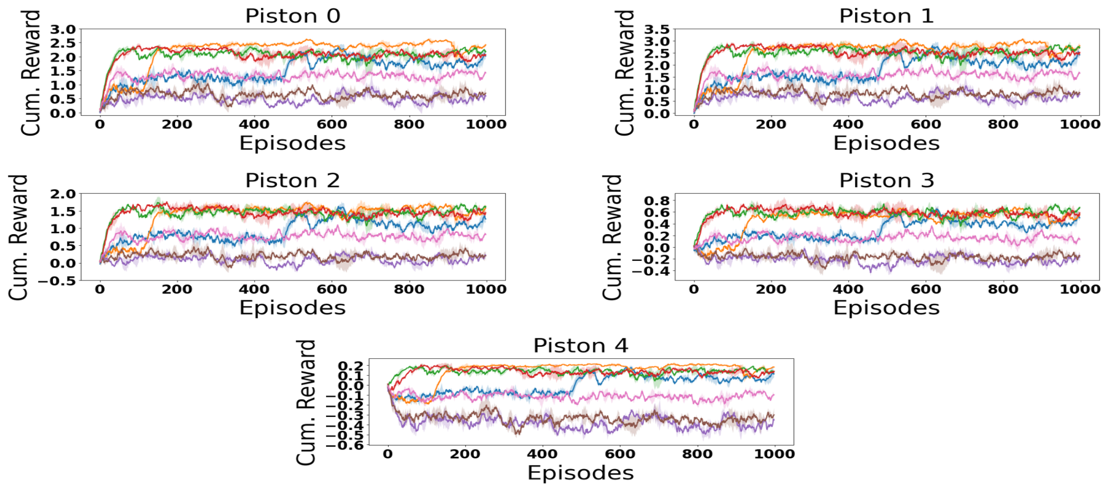
Table D.4: StarCraft II Mini-game (The 3M Challenge) Training Hyperparameters.

Experiments	**StarCraft II (3M Challenge)**					
	InfoPG	**Adv. InfoPG**	**NC-A2C**	**CU**	**MOA**	**PR2-AC**
Learning Rate	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4
Size of Latent Vector	50	50	50	50	50	50
Type of Com. Network	GRU	GRU	-	-	GRU	-
Epochs	1000	1000	1000	1000	1000	1000
MOA Loss Weight	-	-	-	-	0.1	-
Discount Factor	0.99	0.99	0.99	0.99	0.99	0.99
Batch Size	64	64	64	64	64	64
Max Gradient Norm	6	6	6	6	6	6
Replay Buffer Size	-	-	-	-	-	1e5
Number of Particles	-	-	-	-	-	16

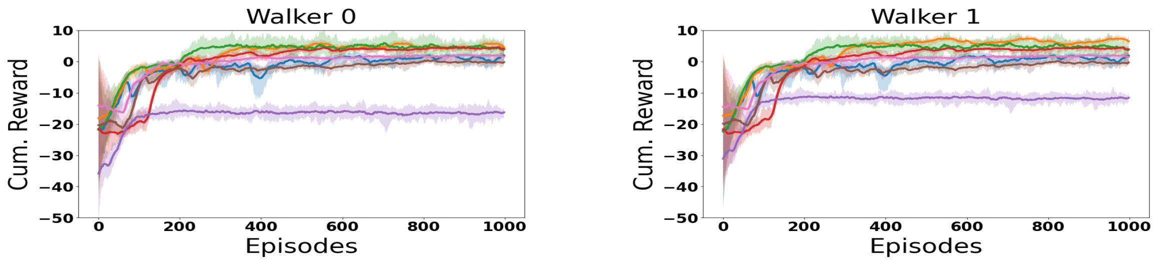
(a) Individual agent performances in Co-op Pong.



(b) Individual agent performances in Pistonball.



(c) Individual agent performances in Multiwalker.



(d) Individual agent performances in StarCraft II (3M).

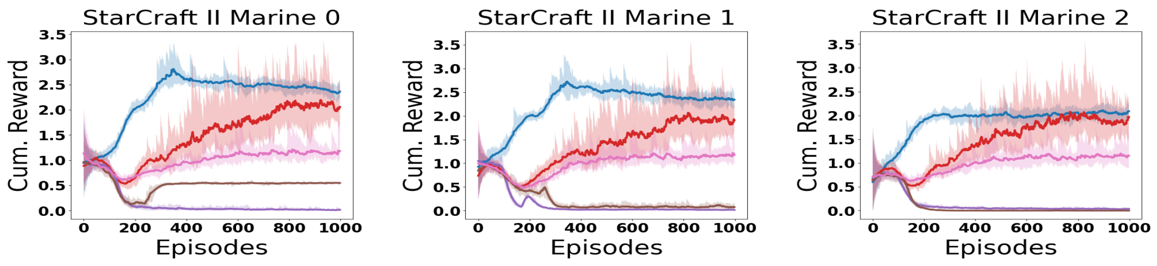


Figure D.6: Individual rewards obtained by each individual agents across episodes as training proceeds in the three evaluation environments. Our Adv. InfoPG continually outperforms all baselines [100, 99, 63, 126] across all domains.

Table D.5: Pistonball Training Hyperparameters for the fraudulent agent experiment.

Experiments	**Fraud Pistonball**		
	InfoPG	**Adv InfoPG**	**MOA**
Learning Rate	1e-3	1e-3	1e-3
Size of Latent Vector	20	20	20
Type of Com. Network	GRU	GRU	GRU
Epochs	1000	1000	1000
MOA Loss Weight	-	-	1.0
Discount Factor	0.99	0.99	0.99
Batch Size	2	2	2
Max Gradient Norm	0.5	0.5	0.5

APPENDIX E
SUPPLEMENTARY INFORMATION (chapter 8)

E.1 Algorithm Details and Pseudocode

In Algorithm algorithm 7, we define MixTURE's training process.

Algorithm 7: MixTURE Training

```
1: For each agent $j$ for each class $c$
   ▷ obtain expert trajectories $\tau_E^{(c_j)}$
   ▷ initialize policies $\pi_\phi^{(c_j)}$ and discriminators $D_\theta^{(c_j)}$.
2: while not converged do
3:     Collect trajectories $\tau = \{(\bar{o}_t, \bar{a}_t)\}_{t=1}^{T}$ by executing policies $\pi_\phi^{(c_j)}$ for each agent $j$,
       class $c$,
4:     storing communications $z_{ij}$ between each pair of agents $i, j$
5:     Predict rewards $r^{(c_j)}(\bar{o}_t, \bar{a}_t) \leftarrow \log(D)^{(c_j)}(\bar{o}_t, \bar{a}_t)$
6:     Update $\pi_\phi^{(c_j)}$ using objective (4)
7:     Train $\mathscr{D}_\theta^{(c_j)}$ to classify expert trajectories $\tau_E$ from collected trajectores $\tau$ using loss
       (1)
8: end while=0
```

We note that we publicly provide our codebase (including MixTURE implementation, the environment implementations, the expert heuristics, and the baselines) at: https://github.com/CORE-Robotics-Lab/MixTURE.

E.2 Proof of Evidence Lower-Bound (ELBO) in Eq. 3

To arrive at the ELBO in Eq. 3, we begin from the MI definition (sketch; to be fixed.):

$$I(\hat{z}_i;\bar{o}) = H(\hat{z}_i) - H(\hat{z}_i|\bar{o}) = \mathbb{E}_{\hat{z}_i\sim\rho(z),a_i\sim\pi_\phi}\Big[\log\big(\rho(\hat{z}_i|\bar{o},a_i)\big)\Big] + H(\hat{z}_i) \tag{E.1}$$

$$= \mathbb{E}_{a_i\sim\pi_\phi}\Bigg[D_{KL}\Big(\log\big(\rho(\hat{z}_i|\bar{o})\big)\,\|\,\log\big(\Theta_i(\hat{z}_i|\bar{o})\big)\Big) + \mathbb{E}_{\hat{z}_i\sim\rho(z)}\big[\Theta_i(\hat{z}_i|\bar{o})\big]\Bigg] + H(\hat{z}_i) \tag{E.2}$$

$$\geq \mathbb{E}_{\hat{z}_i\sim\mathcal{N}(\vec{\mu},\vec{\sigma}^2),a_i\sim\pi_\phi}\Big[\log\big(\Theta_i(\hat{z}_i|\bar{o},a_i)\big)\Big] + H(\hat{z}_i) = L_{MIM}(\pi^i_\phi\,\|\,\Theta_i) \tag{E.3}$$

E.3 Environment Details

E.3.1 Predator-Prey (PP)

The objective within this homogeneous environment is for $\mathcal{N}$ predator agents with limited vision to find a stationary prey and move to its location. The agents in this domain are homogeneous in their state, observation, and action spaces and thus, all agents are of the same *class*. All agents are able to sense/observe the environment and each agent's observation is a concatenated array of the state vectors of all grids within the agent's Field of View (FOV). The predator agents' action-space is of dimension five, including cardinal movements and a null action, and is the same for all agents. A higher-performing algorithm in this domain is defined as one that minimizes the average number of steps taken by agents to complete an episode. Details of the environment setup and problem dimensions are defined in Table Table E.1.

E.3.2 Predator-Capture-Prey (PCP)

In our second domain, we have two classes of agents: *predator* agents and *capture* agents. The first class of agent, called the *predator* agents, have the goal of discovering the prey and have an action-space of dimension five, including cardinal movements and a null (stay)

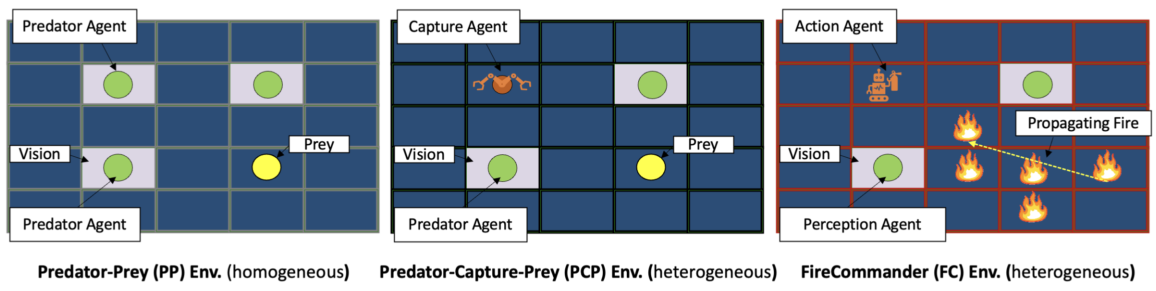


Figure E.1: The PP, PCP, and FC domains utilized for evaluating MixTURE and the baselines.

action. *Predator* agents have an observation space similar to the agents in PP domain. The second class of agents, called the *capture* agents, have the objective of locating the prey *and* capturing it. Capture agents differ from the predator agents in both their observation and their action spaces. Capture agents do not receive any observation inputs from the environment (i.e., no scanning sensors) and have an additional action of *capture-prey* in their action-space. This additional action must be used at a prey's location to capture the prey. Note that this domain is an explicit example of the perception-action composite teams. An episode is deemed successful once all agents have completed their class-specific objectives. Again, a better-performing algorithm in this domain is defined as one that minimizes the average number of steps taken by agents to complete an episode. Details of the environment setup and problem dimensions are defined in Table Table E.1.

Table E.1: Environment Configuration Details

Environment	# predator/perception	# capture/action	vision	max steps
Predator-Prey 5x5	3	-	1x1	20
Predator-Prey 10x10	6	-	3x3	80
Predator-Prey 20x20	10	-	5x5	80
Predator-Capture-Prey 5x5	2	1	1x1	40
Predator-Capture-Prey 10x10	3	3	3x3	80
Predator-Capture-Prey 20 x 20	6	4	5x5	80
FireCommander 5x5	2	1	3x3	80
FireCommander 10x10	3	3	3x3	80
FireCommander 20x20	6	4	5x5	80

E.3.3 FireCommander (FC)

We also evaluate the performance of MixTURE and the baselines in a heterogeneous cooperative multi-agent environment, called FireCommander (FC), recently introduced by [54, 13]. FireCommander can be categorized as a strategic game, in which a composite team of robots (i.e., UAVs) must collaboratively find hidden areas of propagating wildfire and extinguish the fire in such areas as fast as possible. The robot team in FC is composed of two classes of agents: (1) *perception* agents (class P), which can only sense the environment and detect areas of fire and, (2) *action* agents (class A), which can only manipulate the environment by extinguishing a firespot which has already been detected by class P agents. Neither class P, nor class A agents are capable of accomplishing the task on their own, and therefore must communicate and collaborate.

Under the notations in our problem formulation in Section 3, we have $\mathscr{C} = \{\mathrm{P}, \mathrm{A}\}$ where, $\mathscr{A}^{(\mathrm{P})} = \{1, 2, \cdots, 5\}$ representing the four primitive motions and stay (no-op action), and $\mathscr{A}^{(\mathrm{A})} = \{1, \cdots, 6\}$, representing the four primitive motions, stay no-op action, and an extra action which corresponds to extinguishing fire by dousing water. Note that the dimensions of the action space here are similar to the PCP. Agents of class P are equipped with fire detection sensors and can observe the environment, receiving an input vector of length 29 for each grid within their FOV. Agents of class A, do not receive any observation from the environment. An episode of the game is marked as successful only if all the active firespots within the map are discovered and extinguished. Once again, a better-performing algorithm in this domain is defined as one that minimizes the average number of steps taken by agents to complete an episode. Details of the environment setup and problem dimensions are defined in Table Table E.1.

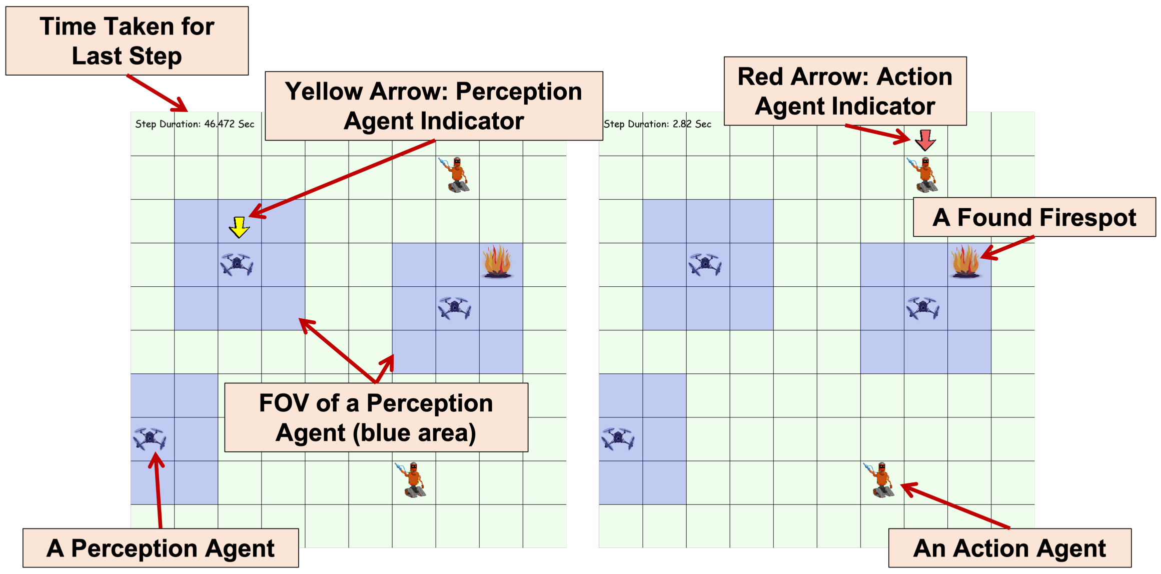

Time Taken for Last Step
Step Duration: 46.472 Sec
Yellow Arrow: Perception Agent Indicator
Step Duration: 2.82 Sec
Red Arrow: Action Agent Indicator
A Found Firespot
FOV of a Perception Agent (blue area)
A Perception Agent
An Action Agent

Figure E.2: Designed FireCommander interface for the human-subject user experiment.

E.4 Human-Subject User Study Details: Environment and Procedures

Here we present further details regarding our experiment setup and environment design for collecting human demonstrations for teaching collaborative policies to multi-robot teams as well as evaluating the effects of demonstrating both environment-action and communication-action strategies on the human expert's performance and system usability. We designed a version of the FireCommander [54, 13] suitable for our experiments. **The following are the detailed instructions and information provided to the human subjects during the experiments**[1].

E.4.1 General Game Objectives and Logistics

In this game, you will have to use a group of Quadcopter robots (a.k.a the Perception agents) and a group of Ground robots (a.k.a the Action agents) to put out a propagating wildfire as fast and efficient as possible. But here are the game logistics:

- Fire is initially hidden from you. You need to search around to find the firespots.

[1]Instructions presented as they were given to the human subjects. 'You' refers to the subject.

- Only Perception agents can see the fire; but they have limited field of view (FOV), only within the blue shaded area around them.
- Only Action agents can put out a firespot; but they do not receive any environment observations (lack of sensory information).
- A firespot keeps propagating and growing in the background on wall-clock time, regardless of what you do, unless you put it out.
- Once all active firespots are put out the game ends and you'll receive a score.
- The faster you put out the fire, the higher you will score.
- Overall, use the quadcopters to find fires, then use ground robots to put out the found fires, and do this as fast and as efficient (fewest # steps taken to finish the game) as possible.
- Note that, to put out a firespot, you must first find it. You cannot put a firespot out before finding it.
- Note that, when you find a fire, its location will be known even if you move away the Perception agent. However, if that fire propagates, you will not see the new grid on fire, unless you have a Perception agent monitoring that area.
- Note that, to put out a firespot, the Action agent must be on top of it (i.e., agents will not be damaged by the fire)
- Note that, the game has an enforced cut off threshold on the score. When your score drops below 50, the game will end automatically.
- Note that, your score is initialized to a 100 and will change based on your strategy and wall-clock time. The score function details are described below.

- Note that, there are three levels to this game: Easy, Moderate, and Hard (see Fig. Figure E.3). You will be practicing on the easy mode for at least one round first, then starting the actual rounds from easy to harder levels.

E.4.2 Understanding the Game Score Function

Your `score` is initialized to 100 at the start of each round and then:

- `score` $-= f_p^s \times$(fixed wall-clock time), where f_p^s is the fire propagation rate for scenario, s, (i.e., easy, moderate, or hard)
- `score` $-= 0.2\times$(per existing firespot)
- `score` $+= 0.2\times$(per found firespot)
- `score` $+= 0.5\times$(per killed firespot)
- `score` $-= 0.1\times$(per usage of fire extinguisher)

E.4.3 Task Description for Each Condition

We utilize a 1×2 within-subjects design varying across two abstractions:

1. ***noComm* condition**: Only demonstrating environment actions for each robot at each time step.
2. ***withComm* condition**: Demonstrating both environment actions and communication actions for each agent at each time step.

We implemented both conditions in the FC interface introduced above. The details of each mode along with user instructions (as were presented to the human subjects during the experiment) for each condition are presented below.

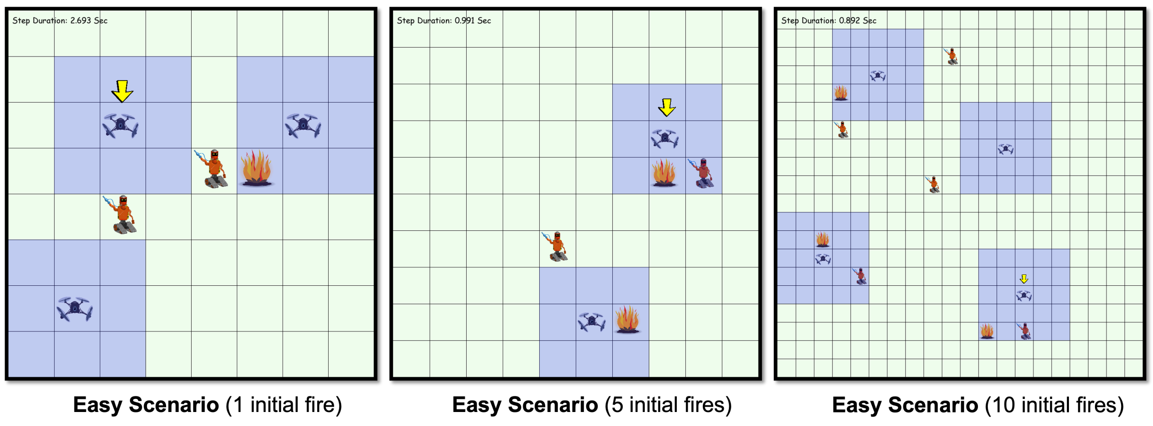

Step Duration: 2.693 Sec
Step Duration: 0.991 Sec
Step Duration: 0.892 Sec
Easy Scenario (1 initial fire)
Easy Scenario (5 initial fires)
Easy Scenario (10 initial fires)

Figure E.3: Instances of the designed FireCommander game environment at different levels for the human-subject user experiment.

Maneuvering the Game in noComm Condition

In this mode, the human subject will be only demonstrating the environment actions for each robot at each time step. The data collected in this condition are used to train our MixTURE architecture. Here are the details:

- At each step of the game, the current agent (i.e., agent that you will be handling) is shown with a downward arrow on top.
- After performing a task, the arrow indicator will move on top of the next agent (turn rotates).
- You will move/handle robots in this way, one-by-one, until the game ends.
- To move the robots around us the arrow keys: **Up** (↑), **Down** (↓), **Right** (→), **Left** (←)
- To remain still (move to next agent w/o doing anything), press the **Enter**
- To put out a fire, use the right or left **Ctrl** key

Maneuvering the Game in withComm Condition)

In this mode, the human subject will be demonstrating both an environment-actions and a communication-action for each robot at each time step. The data collected in this condition are used to train the MA-GAIL baseline [110]. We note that, the messages in the communication menu designed for this condition include a wide range of options from anticipatory (i.e., sharing information by anticipating one another's needs rather than answering to requests or prompts) and deliberative information sharing (i.e., prioritizes information about the next goal to be accomplished during a task), the design of which was inspired by prior work [309, 310]. The dimensionality of this message-space is 26, which is also similar to the dimensions of the message space in our expert heuristic design, as well as the RL-only baseline with learned communication action.

Everything in this mode is just the same as described in *noComm* mode, except that for each robot there is one extra step where you need to select a message from a predefined list to be broadcasted by the current robot to other teammates. Here are the details:

- In this mode, for each agent, first a communication menu will pop up. After message selection is done, the communication menu will be gone and you can move the robot.
- For each agent, you can choose a message from the menu shown below in Fig. Figure E.4 using keyboard numbers.
- Choosing options 1 – 6 would need you to specify your intended area (i.e., domain quadrants by choosing a number 1 – 4).
- Message option 7 indicates a previously incorrect statement by an agent, and message option 8 is a Null message (i.e., no communication).
- Note that, for reference, you can see your game screen (i.e., location of your robots and the fire (if any found)) on the bottom of the communication menu, at all times,

alongside with robot's current partial observation (i.e., what you see v.s. what the robot sees). You should select the communication message based on what robot sees.

- Note that, if you press a wrong key (like the arrow keys), by default, message option 8 (i.e., "Null; nothing important") will be selected.
- Note that, in this case the fire is still propagating on wall-clock time in the background. Try to become familiar with the message options early on and come up with a communication strategy during practice so you won't have to read all the options every time. The communication menu is fixed.
- Important: as stated before, we will use LfD to teach robots how to communicate from your data. Therefore, it is highly important that you accurately select a message that reflects your strategy. Without messages being accurately selected, the robots that will be trained on your data will be inefficient and unsuccessful. Note that consistency is key.

E.5 Expert Heuristic Design Details

E.5.1 Predator-Prey (PP) Expert Heuristic

The heuristic expert behavior for Predator-Prey involves each agent simply moving towards the nearest known prey location. If no prey locations are known, we use a greedy exploration behavior in which each agent attempts to reveal as many unexplored tiles as possible in the next turn. However, to reduce backtracking, we penalize exploring tiles which have unexplored neighbors. Thus, the expert behavior prioritizes exploring corner and edge tiles, and tiles on the boundary of the revealed areas. Table Table E.2 shows the performances achieved by our heuristics in PP, PCP, and FC.

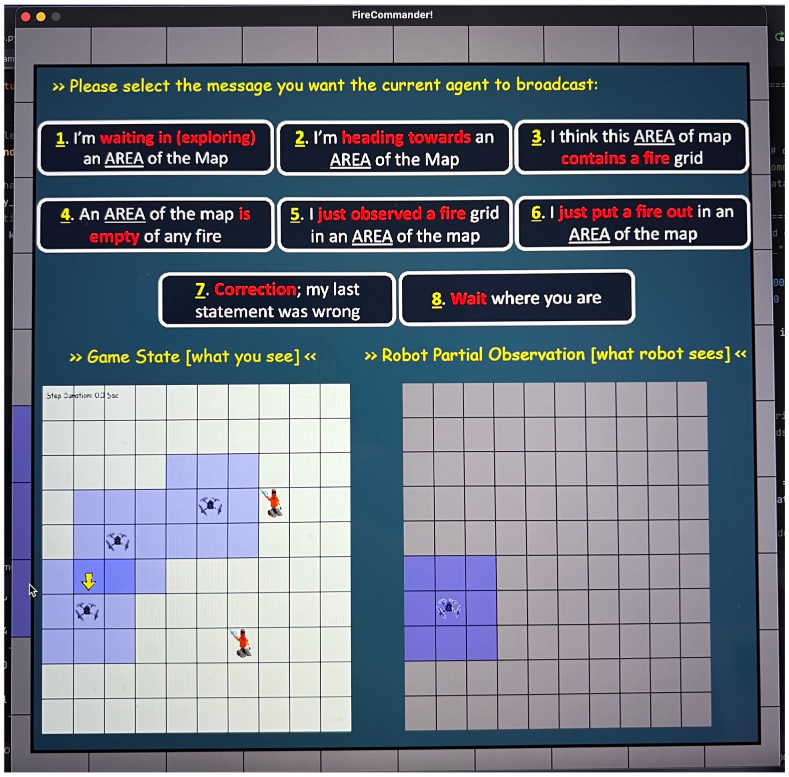

FireCommander!
» Please select the message you want the current agent to broadcast:
1. I'm waiting in (exploring) an AREA of the Map
2. I'm heading towards an AREA of the Map
3. I think this AREA of map contains a fire grid
4. An AREA of the map is empty of any fire
5. I just observed a fire grid in an AREA of the map
6. I just put a fire out in an AREA of the map
7. Correction; my last statement was wrong
8. Wait where you are
» Game State [what you see] «
» Robot Partial Observation [what robot sees] «
Step Duration: 0.1 Sec

Figure E.4: The communication menu designed for expert demonstration in the second condition (i.e., withComm mode). The messages in the communication menu designed for this condition include a wide range of options from anticipatory (i.e., sharing information by anticipating one another's needs rather than answering to requests or prompts) and deliberative information sharing (i.e., prioritizes information about the next goal to be accomplished during a task), the design of which was inspired by prior work [309, 310]. The dimensionality of this message-space is 26.

E.5.2 Predator-Capture-Prey (PCP) Expert Heuristic

Since capture agents find themselves unable to move off of prey, they can effectively see within a (1×1) vision radius by simply checking whether their last attempted move was successful. Thus, we utilize the same logic as the Predator-Prey heuristic, but treat capture agents as predators with a single-cell vision radius. See Table Table E.2 for our heuristics' performance in PP, PCP, and FC.

E.5.3 FireCommander (FC) Expert Heuristic

Instead of tracking revealed and unrevealed tiles, we maintain a probabilistic belief over each grid cell estimating the likelihood that a particular cell contains a fire. These beliefs are

Table E.2: Expert Heuristic Performance (# steps)

Difficulty	Predator-Prey	Predator-Capture-Prey	FireCommander
5×5	8.573 ± 2.175	9.677 ± 2.628	14.439 ± 8.712
10×10	12.221 ± 3.017	14.763 ± 3.858	16.160 ± 8.247
20×20	24.915 ± 5.512	27.701 ± 6.617	24.213 ± 13.721

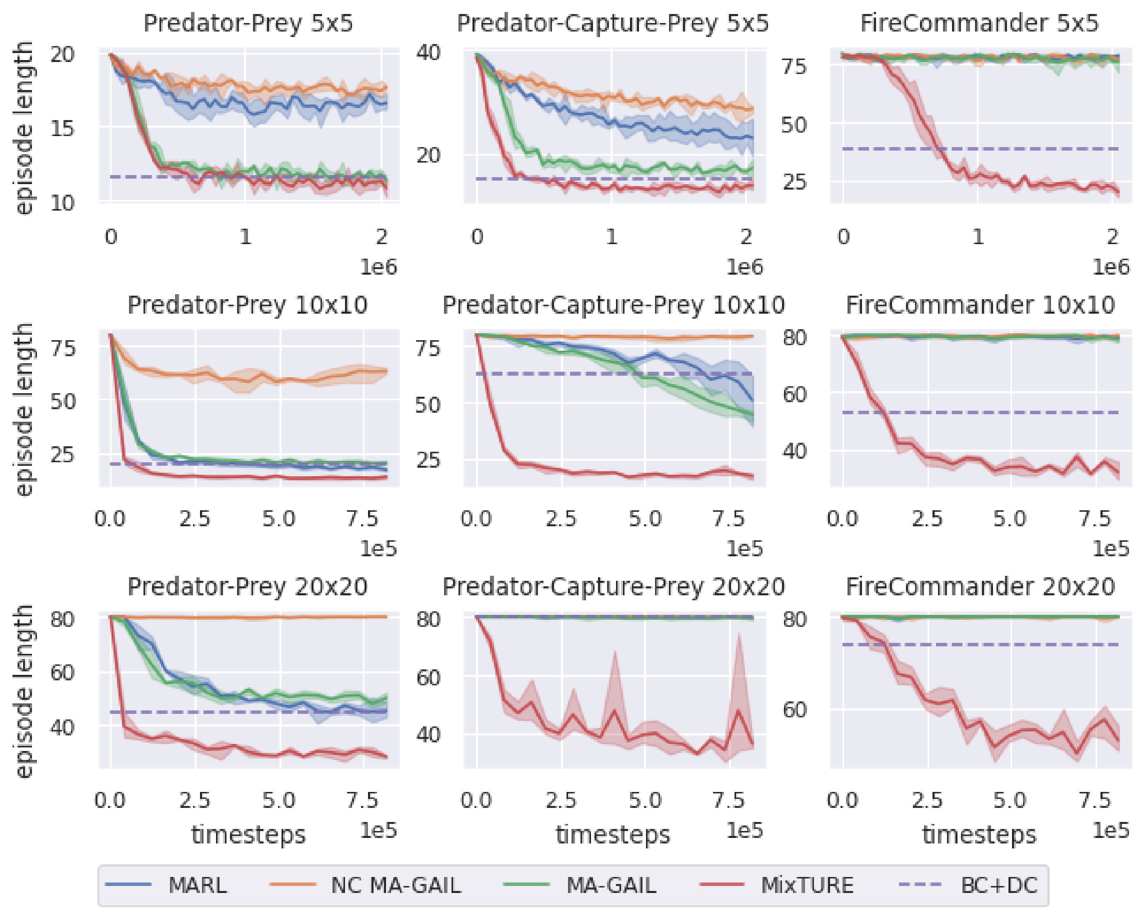


Figure E.5: Full evaluation results for MixTURE and the baselines in the all difficult levels (i.e., easy, moderate, and hard) of the three environments (i.e., PP, PCP, and FC).

updated based on observations from perception agents. To take into account fire spreading dynamics, at each timestep, we increase the estimated fire probabilities according to a Gaussian filter applied to the current belief. The perception agent behavior is similar to that of the Predator-Prey heuristic, but the score of each cell v_{xy} is instead equal to the current entropy of the corresponding belief. Action agents simply attempt to extinguish the closest known fire. If no fires are visible, they instead follow the nearest perception agent.

E.5.4 Communication Heuristic

For the communication heuristic, we used a baseline approach in which each agent generates a one-hot encoded representation (with the same size and dimensionality as the message space available to the human participants) as a function of their last k observations. To embed the observations into a one-hot vector, we simply perform K-means clustering over the observations in the demonstration dataset for each environment, and map each observation to the index of the nearest cluster center. For $k > 1$, we instead perform this procedure over the last k observation vectors, concatenated. We find $k = 2$ to have the best performance, which is in accordance wityh prior work [310].

E.6 Ablation Studies and Supplementary Results

E.6.1 Scalability

We evaluated the MixTURE against all baselines in all three domains introduced in Section section E.5 and under three different difficulty levels: (1) easy (5×5 domain, 3 robots), (2) medium (10×10 domain, 6 robots), and (3) hard (20×20 domain, 10 robots). More environment details are provided in the supplementary material. Fig. Figure E.5 shows the training and evaluation results for MixTURE and the baselines in the medium case, for PP, PCP, and FC domains. Each epoch on the x-axis represents $40K$ data samples. As shown, MixTURE outperforms all non-communicative, communicative with expert heuristic, and communicative with differentiable communication channels including the SOTA MARL framework for learning heterogeneous teaming policies for composite robot teams [13].

E.7 Hyperparameters

Table E.3: Hyperparameters

Name	Value
hidden layer dimensionality	64 (easy), 256 (moderate/hard)
rollout steps	4096
T-BPTT segment length	8
segments per minibatch	8 (easy), 32 (moderate/hard)
total minibatch size	64 (easy), 256 (moderate/hard)
PPO clipping ε	0.2
PPO epochs	3
learning rate	$[10^{-4}, 10^{-3}]$
discount factor	0.99
GAE lambda	0.5
MIM coefficient (λ_{MIM})	0.1 (easy), 0.01 (moderate/hard)
BC coefficient (λ_{BC})	$[10^{-1.5}, 10^{0}]$
discriminator learning rate	10^{-5}
max gradient norm	5.0

REFERENCES

[1] E. Seraj, "Embodied team intelligence in multi-robot systems.", in *AAMAS*, 2022, pp. 1869–1871.

[2] F. Mondada *et al.*, "Swarm-bot: A new distributed robotic concept", *Autonomous robots*, vol. 17, pp. 193–221, 2004.

[3] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: A review from the swarm engineering perspective", *Swarm Intelligence*, vol. 7, pp. 1–41, 2013.

[4] J. S. Albus, A. J. Barbera, R. N. Nagel, *et al.*, *Theory and practice of hierarchical control*. National Bureau of Standards Gaithersburg, MD, USA, 1980.

[5] E. Seraj and M. Gombolay, "Coordinated control of uavs for human-centered active sensing of wildfires", in *2020 American Control Conference (ACC)*, IEEE, 2020, pp. 1845–1852.

[6] E. Seraj, V. Azimi, C. Abdallah, S. Hutchinson, and M. Gombolay, "Adaptive leader-follower control for multi-robot teams with uncertain network structure", in *(ACC 2021)*, IEEE, 2021, pp. 1088–1094.

[7] E. Seraj, A. Silva, and M. Gombolay, "Multi-uav planning for cooperative wildfire coverage and tracking with quality-of-service guarantees", *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 2, p. 39, 2022.

[8] E. Seraj, A. Silva, and M. Gombolay, "Safe coordination of human-robot firefighting teams", *arXiv preprint arXiv:1903.06847*, 2019.

[9] E. Seraj, L. Chen, and M. C. Gombolay, "A hierarchical coordination framework for joint perception-action tasks in composite robot teams", *IEEE Transactions on Robotics*, 2021.

[10] E. Seraj, Z. Wang, R. Paleja, M. Sklar, A. Patel, and M. Gombolay, "Heterogeneous graph attention networks for learning diverse communication", *arXiv preprint arXiv:2108.09568*, 2021.

[11] C. F. Camerer, T.-H. Ho, and J.-K. Chong, "A cognitive hierarchy model of games", *The Quarterly Journal of Economics*, vol. 119, no. 3, pp. 861–898, 2004.

[12] S. Konan, E. Seraj, and M. Gombolay, "Iterated reasoning with mutual information in cooperative and byzantine decentralized teaming", *arXiv preprint arXiv:2201.08484*, 2022.

[13] E. Seraj *et al.*, “Learning efficient diverse communication for cooperative heterogeneous teaming”, in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 2022, pp. 1173–1182.

[14] F. Afghah, A. Razi, J. Chakareski, and J. Ashdown, “Wildfire monitoring in remote areas using autonomous unmanned aerial vehicles”, in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2019, pp. 835–840.

[15] J. V. R. de Sousa and P. V. Gamboa, “Aerial forest fire detection and monitoring using a small uav”, *KnE Engineering*, pp. 242–256, 2020.

[16] Z. Xing, Y. Zhang, C.-Y. Su, Y. Qu, and Z. Yu, “Kalman filter-based wind estimation for forest fire monitoring with a quadrotor uav”, in *2019 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 2019, pp. 783–788.

[17] R. Beard, D. Kingston, T. W. McLain, and D. Nelson, “Decentralized cooperative aerial surveillance using fixed-wing miniature uavs”, 2006.

[18] M. McIntire, E. Nunes, and M. Gini, “Iterated multi-robot auctions for precedence-constrained task scheduling”, in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1078–1086.

[19] E. Nunes and M. Gini, “Multi-robot auctions for allocation of tasks with temporal constraints”, in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[20] H.-L. Choi, L. Brunet, and J. P. How, “Consensus-based decentralized auctions for robust task allocation”, *IEEE transactions on robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[21] R. S. Allison, J. M. Johnston, G. Craig, and S. Jennings, “Airborne optical and thermal remote sensing for wildfire detection and monitoring”, *Sensors*, vol. 16, no. 8, p. 1310, 2016.

[22] F. De Vivo, M. Battipede, P. Gili, A. J. Yezzi, E. Feron, and E. Johnson, “Real-time fire segmentation via active contours for uav integrated wildfire propagation prediction”, in *2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2018, p. 1488.

[23] P. Chamoso, A. Gonzalez-Briones, F. De La Prieta, and J. M. Corchado, “Computer vision system for fire detection and report using uavs.”, in *RSFF*, 2018, pp. 40–49.

[24] L. Merino, F. Caballero, J. R. Martinez-De-Dios, I. Maza, and A. Ollero, "An unmanned aircraft system for automatic forest fire monitoring and measurement", *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 533–548, 2012.

[25] L. Merino, F. Caballero, J. R. M. de Dios, I. Maza, and A. Ollero, "Automatic forest fire monitoring and measurement using unmanned aerial vehicles", in *Proceedings of the 6th International Congress on Forest Fire Research. Edited by DX Viegas. Coimbra, Portugal*, Citeseer, 2010.

[26] C. Yuan, Y. Zhang, and Z. Liu, "A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques", *(CJFR)*, vol. 45, no. 7, pp. 783–792, 2015.

[27] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: Taxonomy and survey", *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2016.

[28] B. Jung and G. S. Sukhatme, "Cooperative multi-robot target tracking", in *Distributed Autonomous Robotic Systems 7*, Springer, 2006, pp. 81–90.

[29] L. Jin, S. Li, H. M. La, X. Zhang, and B. Hu, "Dynamic task allocation in multi-robot coordination for moving target tracking: A distributed approach", *Automatica*, vol. 100, pp. 75–81, 2019.

[30] R. Mottaghi and R. Vaughan, "An integrated particle filter and potential field method applied to cooperative multi-robot target tracking", *Autonomous Robots*, vol. 23, no. 1, pp. 19–35, 2007.

[31] K. Hausman, J. Müller, A. Hariharan, N. Ayanian, and G. S. Sukhatme, "Cooperative multi-robot control for target tracking with onboard sensing", *The International Journal of Robotics Research*, vol. 34, no. 13, pp. 1660–1677, 2015.

[32] R. Bailon-Ruiz and S. Lacroix, "Wildfire remote sensing with uavs: A review from the autonomy point of view", in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2020, pp. 412–420.

[33] P. Sujit, D. Kingston, and R. Beard, "Cooperative forest fire monitoring using multiple uavs", in *Decision and Control, 2007 46th IEEE Conference on*, IEEE, 2007, pp. 4875–4880.

[34] M. Kumar, K. Cohen, and B. Homchaudhuri, "Cooperative control of multiple uninhabited aerial vehicles for monitoring and fighting wildfires", *Journal of Aerospace Computing, Information, and Communication*, vol. 8, no. 1, pp. 1–16, 2011.

[35] K. A. Ghamry and Y. Zhang, "Cooperative control of multiple uavs for forest fire monitoring and detection", in *Mechatronic and Embedded Systems and Applications (MESA), 2016 12th IEEE/ASME International Conference on*, IEEE, 2016, pp. 1–6.

[36] H. X. Pham, H. M. La, D. Feil-Seifer, and M. Deans, "A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking", in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, IEEE, 2017, pp. 6648–6653.

[37] K. Harikumar, J. Senthilnath, and S. Sundaram, "Multi-uav oxyrrhis marina-inspired search and dynamic formation control for forest firefighting", *IEEE Transactions on Automation Science and Engineering*, 2018.

[38] A. Pongpunwattana and R. Rysdyk, "Real-time planning for multiple autonomous vehicles in dynamic uncertain environments", *Journal of Aerospace Computing, Information, and Communication*, vol. 1, no. 12, pp. 580–604, 2004.

[39] S. G. Lee, Y. Diaz-Mercado, and M. Egerstedt, "Multirobot control using time-varying density functions", *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 489–493, 2015.

[40] X. Lin and C. G. Cassandras, "An optimal control approach to the multi-agent persistent monitoring problem in two-dimensional spaces", *IEEE Transactions on Automatic Control*, vol. 60, no. 6, pp. 1659–1664, 2014.

[41] H. X. Pham, H. M. La, D. Feil-Seifer, and M. C. Deans, "A distributed control framework of multiple unmanned aerial vehicles for dynamic wildfire tracking", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[42] W. Li and C. G. Cassandras, "Distributed cooperative coverage control of sensor networks", in *Proceedings of the 44th IEEE Conference on Decision and Control*, IEEE, 2005, pp. 2542–2547.

[43] L. Zuo, M. Yan, Y. Guo, and W. Ma, "An improved kf-rbf based estimation algorithm for coverage control with unknown density function", *Complexity*, vol. 2019, 2019.

[44] M. Santos, S. Mayya, G. Notomista, and M. Egerstedt, "Decentralized minimum-energy coverage control for time-varying density functions", in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, IEEE, 2019, pp. 155–161.

[45] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, "Eyes in the sky: Decentralized control for the deployment of robotic camera networks", 2011.

[46] D. W. Casbeer, D. B. Kingston, R. W. Beard, and T. W. McLain, "Cooperative forest fire surveillance using a team of small unmanned air vehicles", *International Journal of Systems Science*, vol. 37, no. 6, pp. 351–360, 2006.

[47] D. Morvan, "Physical phenomena and length scales governing the behaviour of wildfires", *Fire technology*, vol. 47, no. 2, pp. 437–460, 2011.

[48] Z. Zaidi *et al.*, "Athletic mobile manipulator system for robotic wheelchair tennis", *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2245–2252, 2023.

[49] A. Krishna, Z. Zaidi, L. Chen, R. Paleja, E. Seraj, and M. Gombolay, "Utilizing human feedback for primitive optimization in wheelchair tennis", *arXiv preprint arXiv:2212.14403*, 2022.

[50] E. Pastor, L. Zarate, E. Planas, and J. Arnaldos, "Mathematical models and calculation systems for the study of wildland fire behaviour", *Progress in Energy and Combustion Science*, vol. 29, no. 2, pp. 139–153, 2003.

[51] R. Bailon-Ruiz, S. Lacroix, and A. Bit-Monnot, "Planning to monitor wildfires with a fleet of uavs", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 4729–4734.

[52] H. Ravichandar, K. Shaw, and S. Chernova, "Strata: Unified framework for task assignments in large teams of heterogeneous agents.", *(JAAMAS)*, vol. 34, no. 2, p. 38, 2020.

[53] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: Research challenges", *Ad hoc networks*, vol. 2, no. 4, pp. 351–367, 2004.

[54] E. Seraj, X. Wu, and M. Gombolay, "Firecommander: An interactive, probabilistic multi-agent environment for heterogeneous robot teams", *arXiv preprint arXiv:2011.00165*, 2020.

[55] H. Yuan, H. Ma, and H. Liao, "Coordination mechanism in wireless sensor and actor networks", in *(IMSCCS'06)*, IEEE, vol. 2, 2006, pp. 627–634.

[56] H. Momeni, M. Sharifi, and S. Sedighian, "A new approach to task allocation in wireless sensor actor networks", in *(CICSN)*, IEEE, 2009, pp. 73–78.

[57] K. Sundar, S. Venkatachalam, and S. G. Manyam, "Path planning for multiple heterogeneous unmanned vehicles with uncertain service times", in *(ICUAS)*, IEEE, 2017, pp. 480–487.

[58] N. Sabri *et al.*, "Wireless sensor actor networks", in *(ISWTA)*, IEEE, 2011, pp. 90–95.

[59] S. Guha, K. Munagala, and P. Shi, “Approximation algorithms for restless bandit problems”, *(JACM)*, vol. 58, no. 1, p. 3, 2010.

[60] P. Peng *et al.*, “Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination”, *arXiv:1703.10069*, 2017.

[61] C. Sun, M. Shen, and J. P. How, “Scaling up multiagent reinforcement learning for robotic systems: Learn an adaptive sparse communication graph”, in *2020 (IROS)*, IEEE, 2020, pp. 11 755–11 762.

[62] S. G. Konan, E. Seraj, and M. Gombolay, “Contrastive decision transformers”, in *Conference on Robot Learning*, PMLR, 2023, pp. 2159–2169.

[63] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, “Fully decentralized multi-agent reinforcement learning with networked agents”, in *(ICML)*, 2018, pp. 5872–5881.

[64] R. S. Sutton, “Temporal credit assignment in reinforcement learning.”, 1985.

[65] J. Yang, A. Nakhaei, D. Isele, K. Fujimura, and H. Zha, “Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning”, in *International Conference on Learning Representations*, 2020.

[66] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, “Learning to communicate with deep multi-agent reinforcement learning”, in *Advances in neural information processing systems*, 2016, pp. 2137–2145.

[67] A. Das *et al.*, “Tarmac: Targeted multi-agent communication”, in *International Conference on Machine Learning*, PMLR, 2019, pp. 1538–1546.

[68] S. Sukhbaatar, R. Fergus, *et al.*, “Learning multiagent communication with backpropagation”, in *Advances in Neural Information Processing Systems*, 2016, pp. 2244–2252.

[69] D. Kim *et al.*, “Learning to schedule communication in multi-agent reinforcement learning”, in *International Conference on Learning Representations*, 2018.

[70] J. Foerster *et al.*, “Stabilising experience replay for deep multi-agent reinforcement learning”, in *International conference on machine learning*, PMLR, 2017, pp. 1146–1155.

[71] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani, “Lenient multi-agent deep reinforcement learning”, in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 443–451.

[72] M. Natarajan *et al.*, “Human-robot teaming: Grand challenges”, in *Current Robotics Reports*, 2023.

[73] M. Ghavamzadeh and S. Mahadevan, “Learning to communicate and act using hierarchical reinforcement learning”, *Computer Science Department Faculty Publication Series*, p. 172, 2004.

[74] C. Amato, G. Konidaris, L. P. Kaelbling, and J. P. How, “Modeling and planning with macro-actions in decentralized pomdps”, *Journal of Artificial Intelligence Research*, vol. 64, pp. 817–859, 2019.

[75] K. Zhang, Z. Yang, and T. Başar, “Multi-agent reinforcement learning: A selective overview of theories and algorithms”, *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

[76] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, “Multi-agent game abstraction via graph attention neural network”, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 7211–7218.

[77] Z. Wang and M. Gombolay, “Learning scheduling policies for multi-robot coordination with graph attention networks”, *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4509–4516, 2020.

[78] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks”, *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[79] J. Jiang, C. Dun, T. Huang, and Z. Lu, “Graph convolutional reinforcement learning”, in *International Conference on Learning Representations (ICLR)*, 2020.

[80] D. Adjodah *et al.*, “Communication topologies between learning agents in deep reinforcement learning”, *arXiv preprint arXiv:1902.06740*, 2019.

[81] J. Sheng *et al.*, “Learning structured communication for multi-agent reinforcement learning”, *arXiv preprint arXiv:2002.04235*, 2020.

[82] Y. Niu, R. Paleja, and M. Gombolay, “Multi-agent graph-attention communication and teaming”, in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 2021, pp. 964–973.

[83] M. J. Bays and T. A. Wettergren, “A solution to the service agent transport problem”, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 6443–6450.

[84] S. Q. Zhang, Q. Zhang, and J. Lin, “Efficient communication in multi-agent reinforcement learning via variance based control”, *Advances in Neural Information Processing Systems*, vol. 32, pp. 3235–3244, 2019.

[85] M. Bravo, J. A. Reyes-Ortiz, J. Rodríguez, and B. Silva-López, “Multi-agent communication heterogeneity”, in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2015, pp. 583–588.

[86] S. Xiong, Q. Wu, and Y. Wang, “Distributed coordination of heterogeneous multi-agent systems with output feedback control”, in *(ICUSAI)*, 2019, pp. 106–111.

[87] D. D. R. Meneghetti and R. A. d. C. Bianchi, “Towards heterogeneous multi-agent reinforcement learning with graph neural networks”, *arXiv*, 2020.

[88] C. Frith and U. Frith, “Theory of mind”, *Current biology*, vol. 15, no. 17, R644–R645, 2005.

[89] A. S. Goodie, P. Doshi, and D. L. Young, “Levels of theory-of-mind reasoning in competitive games”, *Journal of Behavioral Decision Making*, vol. 25, no. 1, pp. 95–108, 2012.

[90] G. Tokadlı and M. C. Dorneich, “Interaction paradigms: From human-human teaming to human-autonomy teaming”, in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, IEEE, 2019, pp. 1–8.

[91] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[92] Y. Guan, D. Maity, C. M. Kroninger, and P. Tsiotras, “Bounded-rational pursuit-evasion games”, in *2021 American Control Conference (ACC)*, IEEE, 2021, pp. 3216–3221.

[93] Y. Wen, Y. Yang, and J. Wang, “Modelling bounded rationality in multi-agent interactions by generalized recursive reasoning”, in *IJCAI-20*, 2020, pp. 414–421.

[94] T. Gilovich, D. Griffin, and D. Kahneman, *Heuristics and biases: The psychology of intuitive judgment*. Cambridge university press, 2002.

[95] H. A. Simon, *Models of bounded rationality: Empirically grounded economic reason*. MIT press, 1997, vol. 3.

[96] R. Paleja, M. Ghuy, N. Ranawaka Arachchige, R. Jensen, and M. Gombolay, “The utility of explainable ai in ad hoc human-machine teaming”, *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[97] D. Trendafilov, D. Polani, and R. Murray-Smith, "Model of coordination flow in remote collaborative interaction", in *2015 17th UKSim-AMSS International Conference on Modelling and Simulation (UKSim)*, IEEE, 2015, pp. 361–366.

[98] W. Kim, W. Jung, M. Cho, and Y. Sung, "A maximum mutual information framework for multi-agent reinforcement learning", *arXiv preprint arXiv:2006.02732*, 2020.

[99] N. Jaques *et al.*, *Social influence as intrinsic motivation for multi-agent deep reinforcement learning*, 2019. arXiv: 1810.08647 `[cs.LG]`.

[100] Y. Wen, Y. Yang, R. Luo, J. Wang, and W. Pan, "Probabilistic recursive reasoning for multi-agent reinforcement learning", in *ICLR*, 2019.

[101] S. Chernova and A. L. Thomaz, "Robot learning from human teachers", *Synthesis lectures on artificial intelligence and machine learning*, vol. 8, no. 3, pp. 1–121, 2014.

[102] L. Z. Chen, "Robot learning from heterogeneous demonstration", Ph.D. dissertation, Georgia Institute of Technology, 2020.

[103] J. Ho and S. Ermon, "Generative adversarial imitation learning", *Advances in neural information processing systems*, vol. 29, 2016.

[104] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning", *arXiv preprint arXiv:1710.11248*, 2017.

[105] I. Goodfellow *et al.*, "Generative adversarial networks", *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[106] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation", *arXiv preprint arXiv:1805.01954*, 2018.

[107] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning", in *JMLR*, JMLR, 2011, pp. 627–635.

[108] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning", in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.

[109] H. M. Le, Y. Yue, P. Carr, and P. Lucey, "Coordinated multi-agent imitation learning", in *International Conference on Machine Learning*, PMLR, 2017, pp. 1995–2003.

[110] J. Song, H. Ren, D. Sadigh, and S. Ermon, “Multi-agent generative adversarial imitation learning”, *Advances in neural information processing systems*, vol. 31, 2018.

[111] L. Yu, J. Song, and S. Ermon, “Multi-agent adversarial inverse reinforcement learning”, in *ICML*, PMLR, 2019, pp. 7194–7201.

[112] W. Squires and S. Luke, “Scalable heterogeneous multiagent learning from demonstration”, in *International Conference on Practical Applications of Agents and Multi-Agent Systems*, Springer, 2020, pp. 264–277.

[113] S. G. Subramanian, M. E. Taylor, K. Larson, and M. Crowley, “Multi-agent advisor q-learning”, *Journal of Artificial Intelligence Research*, vol. 74, pp. 1–74, 2022.

[114] R. Hoque *et al.*, “Fleet-dagger: Interactive robot fleet learning with scalable human supervision”, in *Conference on Robot Learning*, PMLR, 2023, pp. 368–380.

[115] J. R. Martinez-de Dios, B. C. Arrue, A. Ollero, L. Merino, and F. Gomez-Rodriguez, “Computer vision techniques for forest fire perception”, *Image and vision computing*, vol. 26, no. 4, pp. 550–562, 2008.

[116] D. Stipaniev, M. tula, L. Krstini Damir and, T. Jakov, and Bugari, “Advanced automatic wildfire surveillance and monitoring network”, in *6th International Conference on Forest Fire Research, Coimbra, Portugal.(Ed. D. Viegas)*, 2010.

[117] K. Fujiwara and J.-i. Kudoh, “Forest fire detection in 2001 using three-dimensional histogram”, in *Geoscience and Remote Sensing Symposium, 2002. IGARSS’02. 2002 IEEE International*, IEEE, vol. 4, 2002, pp. 2057–2059.

[118] J.-I. Kudoh and K. Hosoi, “Two dimensional forest fire detection method by using noaa avhrr images”, in *Geoscience and Remote Sensing Symposium, 2003. IGARSS’03. Proceedings. 2003 IEEE International*, IEEE, vol. 4, 2003, pp. 2494–2495.

[119] R. N. Haksar and M. Schwager, “Distributed deep reinforcement learning for fighting forest fires with a network of aerial robots”, in *(IROS)*, 2018, pp. 1067–1074.

[120] L. F. Oliveira, A. P. Moreira, and M. F. Silva, “Advances in forest robotics: A state-of-the-art survey”, *Robotics*, vol. 10, no. 2, p. 53, 2021.

[121] E. Seraj, X. Wu, and M. C. Gombolay, “Firecommander 2020”, *GitHub Repository, Available Online*, 2020.

[122] M. A. Finney, "Farsite: Fire area simulator-model development and evaluation", *Res. Pap. RMRS-RP-4, Revised 2004. Ogden, UT: US Department of Agriculture, Forest Service, Rocky Mountain Research Station. 47 p.*, vol. 4, 1998.

[123] E. Beachly *et al.*, "Fire-aware planning of aerial trajectories and ignitions", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 685–692.

[124] R. C. Rothermel, *A mathematical model for predicting fire spread in wildland fuels*. Intermountain Forest and Range Experiment Station, Forest Service, US ..., 1972, vol. 115.

[125] M. E. Alexander, "Calculating and interpreting forest fire intensities", *Canadian Journal of Botany*, vol. 60, no. 4, pp. 349–357, 1982.

[126] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[127] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation", *arXiv preprint arXiv:1506.02438*, 2015.

[128] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains", *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[129] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation", *Neural computation*, vol. 3, no. 1, pp. 88–97, 1991.

[130] S. Ross and D. Bagnell, "Efficient reductions for imitation learning", in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[131] A. Y. Ng, S. Russell, *et al.*, "Algorithms for inverse reinforcement learning.", in *Icml*, vol. 1, 2000, p. 2.

[132] L. Chen, S. Jayanthi, R. R. Paleja, D. Martin, V. Zakharov, and M. Gombolay, "Fast lifelong adaptive inverse reinforcement learning from demonstrations", in *Conference on Robot Learning*, PMLR, 2023, pp. 2083–2094.

[133] I. F. Akyildiz and M. C. Vuran, *Wireless sensor networks*. John Wiley & Sons, 2010.

[134] M. Gombolay, R. Wilcox, and J. Shah, "Fast scheduling of multi-robot teams with temporospatial constraints", 2013.

[135] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, "Fast scheduling of robot teams performing tasks with temporospatial constraints", *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 220–239, 2018.

[136] M. C. Gombolay and J. A. Shah, "Schedulability analysis of task sets with upper- and lower-bound temporal constraints", *Journal of Aerospace Information Systems*, vol. 11, no. 12, pp. 821–841, 2014.

[137] A. Ollero and L. Merino, "Unmanned aerial vehicles as tools for forest-fire fighting", *Forest Ecology and Management*, vol. 234, no. 1, S263, 2006.

[138] Y. Fan, L. Chen, and Y. Wang, "Efficient model-free reinforcement learning using gaussian process", *arXiv preprint arXiv:1812.04359*, 2018.

[139] T. Zhou, L. Ding, J. Ji, L. Li, and W. Huang, "Ensemble transform kalman filter (etkf) for large-scale wildland fire spread simulation using farsite tool and state estimation method", *Fire Safety Journal*, vol. 105, pp. 95–106, 2019.

[140] Z. Lin, H. H. Liu, and M. Wotton, "Kalman filter-based large-scale wildfire monitoring with a system of uavs", *IEEE Transactions on Industrial Electronics*, vol. 66, no. 1, pp. 606–615, 2018.

[141] K. D. Julian and M. J. Kochenderfer, "Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning", *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 8, pp. 1768–1778, 2019.

[142] M. C. Gombolay, R. J. Wilcox, A. Diaz, F. Yu, and J. A. Shah, "Towards successful coordination of human and robotic work using automated scheduling tools: An initial pilot study", in *Proc. Robotics: Science and Systems (RSS) Human-Robot Collaboration Workshop (HRC)*, 2013.

[143] M. C. Gombolay, R. A. Gutierrez, S. G. Clarke, G. F. Sturla, and J. A. Shah, "Decision-making authority, team efficiency and human worker satisfaction in mixed human–robot teams", *Autonomous Robots*, vol. 39, no. 3, pp. 293–312, 2015.

[144] M. Gombolay, A. Bair, C. Huang, and J. Shah, "Computational design of mixed-initiative human–robot teaming that considers human factors: Situational awareness, workload, and workflow preferences", *The International journal of robotics research*, vol. 36, no. 5-7, pp. 597–617, 2017.

[145] P. Toth and D. Vigo, *The vehicle routing problem*. SIAM, 2002.

[146] S. Akhlaghi, N. Zhou, and Z. Huang, "Adaptive adjustment of noise covariance in kalman filter for dynamic state estimation", in *2017 IEEE Power & Energy Society General Meeting*, IEEE, 2017, pp. 1–5.

[147] R. Sim, “Stable exploration for bearings-only slam”, in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, IEEE, 2005, pp. 2411–2416.

[148] J. Cortés and M. Egerstedt, “Coordinated control of multi-robot systems: A survey”, *SICE*, vol. 10, no. 6, pp. 495–503, 2017.

[149] P. Delamatar, A. Finley, and C. Babcock, “Downloading and processing noaa hourly weather station data”, *dim (st)*, vol. 1, no. 30538, p. 12, 2013.

[150] S. S. Ge and Y. J. Cui, “New potential functions for mobile robot path planning”, *IEEE Transactions on robotics and automation*, vol. 16, no. 5, pp. 615–620, 2000.

[151] D. Pickem *et al.*, “The robotarium: A remotely accessible swarm robotics research testbed”, in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, 2017, pp. 1699–1706.

[152] J. Allen and B. Walsh, “Enhanced oil spill surveillance, detection and monitoring through the applied technology of unmanned air systems”, in *International oil spill conference*, American Petroleum Institute, vol. 2008, 2008, pp. 113–120.

[153] J. Alonso-Mora, S. Baker, and D. Rus, “Multi-robot formation control and object transport in dynamic environments via constrained optimization”, *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[154] S. Wan, J. Lu, P. Fan, and K. B. Letaief, “Semi-centralized control for multi-robot formation and theoretical lower bound”, *arXiv preprint arXiv:1709.03641*, 2017.

[155] Z. Wang and M. Gombolay, “Learning scheduling policies for multi-robot coordination with graph attention networks”, *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4509–4516, 2020.

[156] Z. Wang and M. Gombolay, “Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints”, in *Robotics: Science and Systems*, 2020.

[157] L. Chaimowicz, T. Sugar, V. Kumar, and M. F. M. Campos, “An architecture for tightly coupled multi-robot cooperation”, in *Proceedings 2001 ICRA. IEEE international conference on robotics and automation (cat. no. 01CH37164)*, IEEE, vol. 3, 2001, pp. 2992–2997.

[158] R. Carelli, C. De la Cruz, and F. Roberti, “Centralized formation control of non-holonomic mobile robots”, *Latin American applied research*, vol. 36, no. 2, pp. 63–69, 2006.

[159] H. Park and S. A. Hutchinson, "Fault-tolerant rendezvous of multirobot systems", *IEEE transactions on robotics*, vol. 33, no. 3, 2017.

[160] X. Chen and R. W. Brockett, "Centralized and decentralized formation control with controllable interaction laws", in *53rd IEEE Conference on Decision and Control*, IEEE, 2014, pp. 601–606.

[161] M. Ji and M. Egerstedt, "Distributed coordination control of multiagent systems while preserving connectedness", *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 693–703, 2007.

[162] Y. Cao, W. Ren, and M. Egerstedt, "Distributed containment control with multiple stationary or dynamic leaders in fixed and switching directed networks", *Automatica*, vol. 48, no. 8, pp. 1586–1597, 2012.

[163] A. Benevento, M. Santos, G. Notarstefano, K. Paynabar, M. Bloch, and M. Egerstedt, "Multi-robot coordination for estimation and coverage of unknown spatial fields", in *2020 International Conference on Robotics and Automation (ICRA)*, 2020.

[164] S. Wan, J. Lu, and P. Fan, "Semi-centralized control for multi robot formation", in *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*, IEEE, 2017, pp. 31–36.

[165] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination", *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[166] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems", *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[167] S. S. Kia, B. Van Scoy, J. Cortes, R. A. Freeman, K. M. Lynch, and S. Martinez, "Tutorial on dynamic average consensus: The problem, its applications, and the algorithms", *IEEE Control Systems Magazine*, vol. 39, no. 3, pp. 40–72, 2019.

[168] Z.-G. Hou, L. Cheng, and M. Tan, "Decentralized robust adaptive control for the multiagent system consensus problem using neural networks", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 3, pp. 636–647, 2009.

[169] G. Chen and F. L. Lewis, "Distributed adaptive tracking control for synchronization of unknown networked lagrangian systems", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 3, pp. 805–816, 2010.

[170] X. Jin, X. Zhao, J. Yu, X. Wu, and J. Chi, “Adaptive fault-tolerant consensus for a class of leader-following systems using neural network learning strategy”, *Neural Networks*, vol. 121, pp. 474–483, 2020.

[171] J. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall, 1991.

[172] V. Azimi, T. Shu, H. Zhao, R. Gehlhar, D. Simon, and A. D. Ames, “Model-based adaptive control of transfemoral prostheses: Theory, simulation, and experiments”, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.

[173] V. Azimi, T. T. Nguyen, M. Sharifi, S. A. Fakoorian, and D. Simon, “Robust ground reaction force estimation and control of lower-limb prostheses: Theory and simulation”, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[174] J. Lai, L. Mejias, and J. J. Ford, “Airborne vision-based collision-detection system”, *Journal of Field Robotics*, vol. 28, no. 2, 2011.

[175] J.-C. Zufferey, A. Beyeler, and D. Floreano, “Autonomous flight at low altitude with vision-based collision avoidance and gps-based path following”, in *ICRA*, IEEE, 2010, pp. 3329–3334.

[176] M. Brenner and B. Nebel, “Continual planning and acting in dynamic multiagent environments”, *Autonomous Agents and Multi-Agent Systems*, vol. 19, no. 3, pp. 297–331, 2009.

[177] C. Undeger and F. Polat, “Multi-agent real-time pursuit”, *Autonomous Agents and Multi-Agent Systems*, vol. 21, no. 1, pp. 69–107, 2010.

[178] H. Ravichandar, K. Shaw, and S. Chernova, “Strata: Unified framework for task assignments in large teams of heterogeneous agents”, *Autonomous Agents and Multi-Agent Systems*, vol. 34, pp. 1–25, 2020.

[179] D. Sarne and B. J. Grosz, “Determining the value of information for collaborative multi-agent planning”, *Autonomous agents and multi-agent systems*, vol. 26, no. 3, pp. 456–496, 2013.

[180] L. Panait and S. Luke, “Cooperative multi-agent learning: The state of the art”, *Autonomous agents and multi-agent systems*, vol. 11, no. 3, pp. 387–434, 2005.

[181] M. N. Prasad and V. R. Lesser, “Learning situation-specific coordination in cooperative multi-agent systems”, *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 2, pp. 173–207, 1999.

[182] M. Freed, W. Fitzgerald, and R. Harris, "Intelligent autonomous surveillance of many targets with few uavs", in *Proceedings of the Research and Development Partnering Conference, Department of Homeland Security, Boston, MA*, 2005.

[183] S. Waharte and N. Trigoni, "Supporting search and rescue operations with uavs", in *2010 International Conference on Emerging Security Technologies*, IEEE, 2010, pp. 142–147.

[184] P. Rudol and P. Doherty, "Human body detection and geolocalization for uav search and rescue missions using color and thermal imagery", in *2008 IEEE aerospace conference*, Ieee, 2008, pp. 1–8.

[185] T. Tomic *et al.*, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue", *IEEE robotics and automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[186] M. A. Olivares-Mendez *et al.*, "Towards an autonomous vision-based unmanned aerial system against wildlife poachers", *Sensors*, vol. 15, no. 12, pp. 31 362–31 391, 2015.

[187] L. F. Gonzalez, G. A. Montes, E. Puig, S. Johnson, K. Mengersen, and K. J. Gaston, "Unmanned aerial vehicles (uavs) and artificial intelligence revolutionizing wildlife monitoring and conservation", *Sensors*, vol. 16, no. 1, p. 97, 2016.

[188] E. Bondi *et al.*, "Near real-time detection of poachers from drones in airsim.", in *IJCAI*, 2018, pp. 5814–5816.

[189] V. V. Klemas, "Coastal and environmental remote sensing from unmanned aerial vehicles: An overview", *Journal of coastal research*, vol. 31, no. 5, pp. 1260–1267, 2015.

[190] M. M. Marques *et al.*, "Oil spills detection: Challenges addressed in the scope of the seagull project", in *OCEANS 2016 MTS/IEEE Monterey*, IEEE, 2016, pp. 1–6.

[191] C. C. Haddal and J. Gertler, "Homeland security: Unmanned aerial vehicles and border surveillance", Library of Congress Washington DC Congressional Research Service, 2010.

[192] P. D. Vascik, H. Balakrishnan, and R. J. Hansman, "Assessment of air traffic control for urban air mobility and unmanned systems", 2018.

[193] D. P. Thipphavong *et al.*, "Urban air mobility airspace integration concepts and considerations", in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3676.

[194] P. D. Vascik and R. J. Hansman, “Scaling constraints for urban air mobility operations: Air traffic control, ground infrastructure, and noise”, in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3849.

[195] J. Xiao, G. Wang, Y. Zhang, and L. Cheng, “A distributed multi-agent dynamic area coverage algorithm based on reinforcement learning”, *IEEE Access*, vol. 8, pp. 33 511–33 521, 2020.

[196] A. A. Adepegba, S. Miah, and D. Spinello, “Multi-agent area coverage control using reinforcement learning”, in *The Twenty-Ninth International Flairs Conference*, 2016.

[197] R. Zanol, F. Chiariotti, and A. Zanella, “Drone mapping through multi-agent reinforcement learning”, in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, 2019, pp. 1–7.

[198] E. Seraj, “Embodied team intelligence in multi-robot systems”, in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 2022, pp. 1869–1871.

[199] K. D. Julian and M. J. Kochenderfer, “Autonomous distributed wildfire surveillance using deep reinforcement learning”, in *2018 AIAA Guidance, Navigation, and Control Conference*, 2018, p. 1589.

[200] A. Viseras, M. Meissner, and J. Marchal, “Wildfire front monitoring with multiple uavs using deep q-learning”, *IEEE Access*, 2021.

[201] N. K. Ure, S. Omidshafiei, B. T. Lopez, A.-a. Agha-Mohammadi, J. P. How, and J. Vian, “Online heterogeneous multiagent learning under limited communication with applications to forest fire management”, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 5181–5188.

[202] R. E. Kalman and R. S. Bucy, “New results in linear filtering and prediction theory”, *Journal of basic engineering*, vol. 83, no. 1, pp. 95–108, 1961.

[203] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[204] X. Wang, B. Golden, and E. Wasil, “A steiner zone variable neighborhood search heuristic for the close-enough traveling salesman problem”, *Computers & Operations Research*, vol. 101, pp. 200–219, 2019.

[205] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The traveling salesman problem: a computational study*. Princeton university press, 2006.

[206] L. Kraemer and B. Banerjee, “Multi-agent reinforcement learning as a rehearsal for decentralized planning”, *Neurocomputing*, vol. 190, pp. 82–94, 2016.

[207] E. Seraj, v. Azimi, C. Abdallah, S. Hutchinson, and M. Gombolay, “Adaptive leader-follower control for multi-robot teams with uncertain network structure”, in *2021 American Control Conference (ACC)*, IEEE, 2021.

[208] G. Danoy, M. R. Brust, and P. Bouvry, “Connectivity stability in autonomous multi-level uav swarms for wide area monitoring”, in *Proceedings of the 5th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications*, ACM, 2015, pp. 1–8.

[209] M. Li, K. Lu, H. Zhu, M. Chen, S. Mao, and B. Prabhakaran, “Robot swarm communication networks: Architectures, protocols, and applications”, in *2008 Third International Conference on Communications and Networking in China*, IEEE, 2008, pp. 162–166.

[210] Y. Rizk, M. Awad, and E. W. Tunstel, “Cooperative heterogeneous multi-robot systems: A survey”, *(CSUR)*, vol. 52, no. 2, p. 29, 2019.

[211] G. A. Korsah, A. Stentz, and M. B. Dias, “A comprehensive taxonomy for multi-robot task allocation”, *(IJRR)*, vol. 32, no. 12, pp. 1495–1512, 2013.

[212] T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyildiz, “A distributed coordination framework for wireless sensor and actor networks”, in *(ACM Mobihoc)*, ACM, 2005, pp. 99–110.

[213] P. Whittle, “Restless bandits: Activity allocation in a changing world”, *Journal of Applied Probability*, vol. 25, no. A, pp. 287–298, 1988.

[214] K. Akkaya, A. Thimmapuram, F. Senel, and S. Uludag, “Distributed recovery of actor failures in wireless sensor and actor networks”, in *(WCNC)*, IEEE, 2008, pp. 2480–2485.

[215] A. Alfadhly, U. Baroudi, and M. Younis, “Least distance movement recovery approach for large scale wireless sensor and actor networks”, in *(IWCMC)*, IEEE, 2011, pp. 2058–2063.

[216] F. Xia, Y.-C. Tian, Y. Li, and Y. Sung, “Wireless sensor/actuator network design for mobile control applications”, *Sensors*, vol. 7, no. 10, pp. 2157–2173, 2007.

[217] L. Merino, F. Caballero, J. Martinez-de Dios, and A. Ollero, “Cooperative fire detection using unmanned aerial vehicles”, in *(ICRA)*, IEEE, 2005, pp. 1884–1889.

[218] L. Merino, F. Caballero, J. R. Martinez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple UAVs: Application to automatic detection of forest fires", *(JFR)*, vol. 23, no. 3-4, pp. 165–184, 2006.

[219] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating UAVs", in *Cooperative control: models, applications and algorithms*, Springer, 2003, pp. 23–41.

[220] M. Y. Sir, I. F. Senturk, E. Sisikoglu, and K. Akkaya, "An optimization-based approach for connecting partitioned mobile sensor/actuator networks", in *(INFOCOM WKSHPS)*, IEEE, 2011, pp. 525–530.

[221] Y.-L. Lai and J.-R. Jiang, "Optimal path planning for fault-tolerant and energy-efficient target surveillance in wireless sensor and actor networks", in *(MDM: Systems, Services and Middleware)*, IEEE, 2009, pp. 531–535.

[222] J. Le Ny, M. Dahleh, and E. Feron, "Multi-UAV dynamic routing with partial observations using restless bandit allocation indices", in *(ACC)*, IEEE, 2008, pp. 4220–4225.

[223] M. Baykal-Gürsoy, "Semi-markov decision processes", *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[224] D. Bertsimas and J. Niño-Mora, "Restless bandits, linear programming relaxations, and a primal-dual index heuristic", *Operations Research*, vol. 48, no. 1, pp. 80–90, 2000.

[225] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning", in *(ICML)*, 2016, pp. 1928–1937.

[226] T. Haarnoja *et al.*, "Soft actor-critic algorithms and applications", *arXiv preprint arXiv:1812.05905*, 2018.

[227] Y. Hsu, H. Wu, K. You, and S. Song, "A selected review on reinforcement learning based control for autonomous underwater vehicles", *arXiv preprint arXiv:1911.11991*, 2019.

[228] H. X. Pham, H. M. La, D. Feil-Seifer, and L. Van Nguyen, "Reinforcement learning for autonomous UAV navigation using function approximation", in *(SSRR)*, IEEE, 2018, pp. 1–6.

[229] G. Welch, G. Bishop, *et al.*, "An introduction to the Kalman filter", 1995.

[230] B. Ristic, S. Arulampalam, and N. Gordon, "Beyond the kalman filter", *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 7, pp. 37–38, 2004.

[231] P. Newman, *Ekf based navigation and slam, slam summer school 2006*, 2006.

[232] M. Reinštein, “From bayes to extended kalman filter”,

[233] M. Di, E. M. Joo, and L. H. Beng, “A comprehensive study of kalman filter and extended kalman filter for target tracking in wireless sensor networks”, in *(IEEE SMC)*, IEEE, 2008, pp. 2792–2797.

[234] E. Masazade, M. Fardad, and P. K. Varshney, “Sparsity-promoting extended kalman filtering for target tracking in wireless sensor networks”, *(IEEE SPL)*, vol. 19, no. 12, pp. 845–848, 2012.

[235] D. J. MacKay, “Information-based objective functions for active data selection”, *Neural Computation*, vol. 4, no. 4, pp. 590–604, 1992.

[236] P. Whaite and F. P. Ferrie, “Autonomous exploration: Driven by uncertainty”, *(IEEE TPAMI)*, vol. 19, no. 3, pp. 193–205, 1997.

[237] L. Busoniu, R. Babuska, and B. De Schutter, “A comprehensive survey of multiagent reinforcement learning”, *IEEE TSMC*, vol. 38, no. 2, pp. 156–172, 2008.

[238] E. Seraj *et al.*, “Heterogeneous policy networks for composite robot team communication and coordination”, in *The international Journal of Robotics Research (IJRR)*, 2023.

[239] J. MacMillan, E. E. Entin, and D. Serfaty, “Communication overhead: The hidden cost of team cognition.”, 2004.

[240] J. E. Mathieu, T. S. Heffner, G. F. Goodwin, E. Salas, and J. A. Cannon-Bowers, “The influence of shared mental models on team process and performance.”, *Journal of applied psychology*, vol. 85, no. 2, p. 273, 2000.

[241] E. Salas, T. L. Dickinson, S. A. Converse, and S. I. Tannenbaum, “Toward an understanding of team performance and training.”, 1992.

[242] H. Taylor, “The effects of interpersonal communication style on task performance and well being”, Ph.D. dissertation, University of Buckingham, 2007.

[243] L. Busoniu, R. Babuka, and B. D. Schutter, “A comprehensive survey of multiagent reinforcement learning”, *IEEE TSMC*, vol. 38, pp. 156–172, 2008.

[244] C. Zhang and V. R. Lesser, “Coordinating multi-agent reinforcement learning with limited communication.”, in *International Conference on Autonomous Agents and Multiagent Systems*, 2013, pp. 1101–1108.

[245] C. Yu, X. Wang, and Z. Feng, “Coordinated multiagent reinforcement learning for teams of mobile sensing robots”, in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019, pp. 2297–2299.

[246] J. M. Catacora Ocana, F. Riccio, R. Capobianco, and D. Nardi, “Cooperative multi-agent deep reinforcement learning in soccer domains”, in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019, pp. 1865–1867.

[247] O. Vinyals *et al.*, “Grandmaster level in starcraft ii using multi-agent reinforcement learning”, *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[248] C. Berner *et al.*, “Dota 2 with large scale deep reinforcement learning”, *arXiv preprint arXiv:1912.06680*, 2019.

[249] Y. Du *et al.*, “Learning correlated communication topology in multi-agent reinforcement learning”, in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 2021, pp. 456–464.

[250] H. Mao, Z. Gong, Y. Ni, and Z. Xiao, “Accnet: Actor-coordinator-critic net for" learning-to-communicate" with deep multi-agent reinforcement learning”, *arXiv preprint arXiv:1706.03235*, 2017.

[251] A. Singh, T. Jain, and S. Sukhbaatar, “Learning when to communicate at scale in multiagent cooperative and competitive tasks”, *arXiv:1812.09755*, 2018.

[252] E. Pesce and G. Montana, “Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication”, *Machine Learning*, pp. 1–21, 2020.

[253] J. Jiang and Z. Lu, “Learning attentional communication for multi-agent cooperation”, *Advances in Neural Information Processing Systems*, vol. 31, pp. 7254–7264, 2018.

[254] T. Chu, S. Chinchali, and S. Katti, “Multi-agent reinforcement learning for networked system control”, in *International Conference on Learning Representations*, 2019.

[255] X. Xu, R. Li, Z. Zhao, and H. Zhang, “Stigmergic independent reinforcement learning for multiagent collaboration”, *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[256] L. Pimentel, R. Paleja, Z. Wang, E. Seraj, J. E. Pagan, and M. Gombolay, “Scaling multi-agent reinforcement learning via state upsampling”,

[257] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains”, *Artif. Intell.*, vol. 101, pp. 99–134, 1998.

[258] I. Grondman, L. Buşoniu, G. A. D. Lopes, and R. Babuka, “A survey of actor-critic reinforcement learning: Standard and natural policy gradients”, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, pp. 1291–1307, 2012.

[259] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, “Reinforcement learning through asynchronous advantage actor-critic on a GPU”, in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[260] G. Tesauro, “Temporal difference learning and td-gammon”, *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[261] R. S. Sutton, “Temporal credit assignment in reinforcement learning”, Ph.D. dissertation, University of Massachusetts Amherst, 1984.

[262] J. Zhou *et al.*, “Graph neural networks: A review of methods and applications”, *AI Open*, vol. 1, pp. 57–81, 2020.

[263] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph Attention Networks”, *International Conference on Learning Representations*, 2018.

[264] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumbel-softmax”, *arXiv preprint arXiv:1611.01144*, 2016.

[265] S. Levine, “Policy gradients”, *CS 294-112: Deep Reinforcment Learning*, 2018.

[266] S. W. Smith *et al.*, “The scientist and engineer’s guide to digital signal processing”, 1997.

[267] N. Malakar, K. Knuth, and D. Lary, “Maximum joint entropy and information-based collaboration of automated learning machines”, in *AIP Conference Proceedings 31st*, American Institute of Physics, vol. 1443, 2012, pp. 230–237.

[268] T. Wang, J. Wang, Y. Wu, and C. Zhang, “Influence-based multi-agent exploration”, *CoRR*, vol. abs/1910.05512, 2019. arXiv: 1910.05512.

[269] T.-H. Ho and X. Su, “A dynamic level-k model in sequential games”, *Management Science*, vol. 59, no. 2, pp. 452–469, 2013.

[270] F. S. Melo, M. T. Spaan, and S. J. Witwicki, “Querypomdp: Pomdp-based communication in multiagent systems”, in *European Workshop on Multi-Agent Systems*, Springer, 2011, pp. 189–204.

[271] A. Kraskov, H. Stögbauer, and P. Grassberger, “Estimating mutual information”, *Physical review E*, vol. 69, no. 6, p. 066 138, 2004.

[272] B. Poole, S. Ozair, A. van den Oord, A. A. Alemi, and G. Tucker, *On variational bounds of mutual information*, 2019. arXiv: 1905.06922 `[cs.LG]`.

[273] R. B. Myerson, *Game theory*. Harvard university press, 2013.

[274] R. Nagel, “Unraveling in guessing games: An experimental study”, *The American Economic Review*, vol. 85, no. 5, pp. 1313–1326, 1995.

[275] M. Fellows, A. Mahajan, T. G. Rudner, and S. Whiteson, “Virel: A variational inference framework for reinforcement learning”, *Advances in Neural Information Processing Systems*, vol. 32, pp. 7122–7136, 2019.

[276] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson, “Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning”, in *International Conference on Learning Representations*, 2018.

[277] S. Prasad, “Bayesian error-based sequences of statistical information bounds”, *IEEE Transactions on Information theory*, vol. 61, no. 9, pp. 5052–5062, 2015.

[278] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem”, in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 203–226.

[279] J. Peng, W. Li, and Q. Ling, “Byzantine-robust decentralized stochastic optimization over static and time-varying networks”, *Signal Processing*, vol. 183, p. 108 020, 2021.

[280] Z. Allen-Zhu, F. Ebrahimianghazani, J. Li, and D. Alistarh, “Byzantine-resilient non-convex stochastic gradient descent”, in *International Conference on Learning Representations*, 2020.

[281] J. J. Ruel and M. P. Ayres, “Jensen’s inequality predicts effects of environmental variation”, *Trends in Ecology & Evolution*, vol. 14, no. 9, pp. 361–366, 1999.

[282] J. K. Terry *et al.*, “Pettingzoo: Gym for multi-agent reinforcement learning”, *arXiv preprint arXiv:2009.14471*, 2020.

[283] J. K. Gupta, M. Egorov, and M. Kochenderfer, “Cooperative multi-agent control using deep reinforcement learning”, in *International Conference on Autonomous Agents and Multiagent Systems*, Springer, 2017, pp. 66–83.

[284] O. Vinyals *et al.*, “Starcraft ii: A new challenge for reinforcement learning”, *arXiv preprint arXiv:1708.04782*, 2017.

[285] L. Chen, R. Paleja, and M. Gombolay, “Learning from suboptimal demonstration via self-supervised reward regression”, in *CoRL*, 2021, pp. 1262–1277.

[286] D. Abel *et al.*, “On the expressivity of markov reward”, *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[287] L. Sanneman and J. Shah, “Transparent value alignment”, in *Companion of the 2023 ACM/IEEE International Conference on Human-Robot Interaction*, 2023, pp. 557–560.

[288] S. Schaal, “Learning from demonstration”, *Advances in neural information processing systems*, vol. 9, 1996.

[289] E. Seraj, J. Xiong, M. Schrum, and M. Gombolay, “Mixed-initiative multiagent apprenticeship learning for human training of robot teams”, in *Conference on Robot Learning*, 2023.

[290] J. Clark and D. Amodei, “Faulty reward functions in the wild”, *Internet: https://blog. openai. com/faulty-reward-functions*, 2016.

[291] S. M. Kakade, *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

[292] N. Gruver, J. Song, M. J. Kochenderfer, and S. Ermon, “Multi-agent adversarial inverse reinforcement learning with latent variables”, in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1855–1857.

[293] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments”, *Advances in neural information processing systems*, vol. 30, 2017.

[294] J. Hu, M. P. Wellman, *et al.*, “Multiagent reinforcement learning: Theoretical framework and an algorithm.”, in *ICML*, vol. 98, 1998, pp. 242–250.

[295] A. Gautam and S. Mohan, “A review of research in multi-robot systems”, in *2012 IEEE 7th international conference on industrial and information systems (ICIIS)*, IEEE, 2012, pp. 1–5.

[296] G. Hoffman and C. Breazeal, “Collaboration in human-robot teams”, in *AIAA 1st intelligent systems technical conference*, 2004, p. 6434.

[297] M. Bettini, A. Shankar, and A. Prorok, “Heterogeneous multi-robot reinforcement learning”, *arXiv preprint arXiv:2301.07137*, 2023.

[298] M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning”, in *Machine learning proceedings 1994*, Elsevier, 1994, pp. 157–163.

[299] A. Tung *et al.*, “Learning multi-arm manipulation through collaborative teleoperation”, in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 9212–9219.

[300] D. Khurana, A. Koli, K. Khatter, and S. Singh, “Natural language processing: State of the art, current trends and challenges”, *Multimedia tools and applications*, pp. 1–32, 2022.

[301] I. A. Hameed, “Using natural language processing (nlp) for designing socially intelligent robots”, in *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, IEEE, 2016, pp. 268–269.

[302] M. L. Cummings and C. E. Nehme, “Modeling the impact of workload in network centric supervisory control settings”, in *Neurocognitive and physiological factors during high-tempo operations*, CRC Press, 2018, pp. 23–41.

[303] B. Donmez, C. Nehme, and M. L. Cummings, “Modeling workload impact in multiple unmanned vehicle supervisory control”, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 40, no. 6, pp. 1180–1190, 2010.

[304] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms”, *arXiv preprint arXiv:1707.06347*, 2017.

[305] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “Infogan: Interpretable representation learning by information maximizing generative adversarial nets”, in *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*, 2016, pp. 2180–2188.

[306] R. Paleja, A. Silva, L. Chen, and M. Gombolay, “Interpretable and personalized apprenticeship scheduling: Learning interpretable scheduling policies from heterogeneous user demonstrations”, *Advances in Neural Information Processing Systems*, vol. 33, pp. 6417–6428, 2020.

[307] L. Fischer, B. Hammer, and H. Wersing, "Combining offline and online classifiers for life-long learning", in *2015 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2015, pp. 1–8.

[308] A. Steinfeld, O. C. Jenkins, and B. Scassellati, "The oz of wizard: Simulating the human for interaction research", in *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, 2009, pp. 101–108.

[309] A. Butchibabu, C. Sparano-Huiban, L. Sonenberg, and J. Shah, "Implicit coordination strategies for effective team communication", *Human factors*, vol. 58, no. 4, pp. 595–610, 2016.

[310] A. Butchibabu, "Anticipatory communication strategies for human robot team coordination", Ph.D. dissertation, Massachusetts Institute of Technology, 2016.

[311] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research", in *Advances in psychology*, vol. 52, Elsevier, 1988, pp. 139–183.

[312] J. Brooke *et al.*, "Sus-a quick and dirty usability scale", *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

[313] C. Jiang, Z. Chen, and Y. Guo, "Multi-robot formation control: A comparison between model-based and learning-based methods", *Journal of Control and Decision*, vol. 7, no. 1, pp. 90–108, 2020.

[314] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[315] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library", in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[316] M. Wang *et al.*, "Deep graph library: A graph-centric, highly-performant package for graph neural networks", *arXiv preprint arXiv:1909.01315*, 2019.

[317] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation", *CoRR*, vol. abs/1308.3432, 2013. arXiv: 1308.3432.

[318] F. Karimzadeh, "Hardware-friendly model compression techniques for deep learning accelerators", Ph.D. dissertation, Georgia Institute of Technology, 2022.

[319] F. Karimzadeh and A. Raychowdhury, “Towards energy efficient dnn accelerator via sparsified gradual knowledge distillation”, in *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*, IEEE, 2022, pp. 1–6.

[320] F. Karimzadeh, N. Cao, B. Crafton, J. Romberg, and A. Raychowdhury, “Hardware-aware pruning of dnns using lfsr-generated pseudo-random indices”, in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2020, pp. 1–5.

[321] F. Karimzadeh, J.-H. Yoon, and A. Raychowdhury, “Bits-net: Bit-sparse deep neural network for energy-efficient rram-based compute-in-memory”, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 5, pp. 1952–1961, 2022.

[322] F. Karimzadeh, R. Boostani, E. Seraj, and R. Sameni, “A distributed classification procedure for automatic sleep stage scoring based on instantaneous electroencephalogram phase and envelope features”, *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 2, pp. 362–370, 2017.

[323] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, “Natural actor–critic algorithms”, *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.

[324] V. S. Borkar, “Stochastic approximation with two time scales”, *Systems & Control Letters*, vol. 29, no. 5, pp. 291–294, 1997.

[325] J. N. Tsitsiklis and B. Van Roy, “An analysis of temporal-difference learning with function approximation”, *IEEE transactions on automatic control*, vol. 42, no. 5, pp. 674–690, 1997.

[326] S. J. Gershman, E. J. Horvitz, and J. B. Tenenbaum, “Computational rationality: A converging paradigm for intelligence in brains, minds, and machines”, *Science*, vol. 349, no. 6245, pp. 273–278, 2015.

[327] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning”, in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[328] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.