Coordination Strategies for Human-Autonomy Teaming: Learning, Interpretation, and Adaptation

by

Haochen Wu

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering and Scientific Computing)
in The University of Michigan
2024

Doctoral Committee:

Professor Bogdan I. Epureanu, Chair
Professor Kira L. Barton
Professor Alparslan Emrah Bayrak, Lehigh University
Professor Tulga Ersal
Research Professor Amin Ghadami, University of Southern California
Professor Bogdan I. Popa

Haochen Wu
haochenw@umich.edu
ORCID iD: 0000-0001-5252-6707

© Haochen Wu 2024

DEDICATION

To my family.

ACKNOWLEDGMENTS

My Ph.D. studies at the University of Michigan have been both challenging and rewarding. This journey would not have been possible without the support of many individuals. First and foremost, I would like to sincerely thank my Ph.D. advisor, Prof. Bogdan Epureanu, for his guidance and empowerment throughout this journey. It all began with an email bearing the subject line with one word "Opportunity". Thank you for consistently bringing abstract ideas into realistic examples and inspiring me with your innovative and transformative mindset. I would also like to thank my doctoral committee members, Prof. Kira Barton, Prof. Alparslan Emrah Bayrak, Prof. Tulga Ersal, Prof. Amin Ghadami, and Prof. Bogdan Popa for their insights and suggestions for this dissertation. I am especially thankful to Prof. Amin Ghadami for his mentorship at the beginning of my journey. Moreover, I am incredibly grateful to all the members and friends of the Epureanu Research Group for the time we have spent together both inside and outside the lab. Additionally, I have been fortunate to collaborate with Connor Esterwood, Tim Ohtake, Dr. Jessie Cossitt, and Himanshu Saxena to make the human subject study possible. I am also deeply appreciative of my two AI research internships: one at SRI International, advised by Dr. Pedro Sequeira, and one at JPMorgan AI Research, supervised by Dr. Sriram Gopalakrishnan. These experiences helped me realize the scalable impact I could make with the skills earned throughout the Ph.D. journey. Also, I am thankful for the research support provided by the Automotive Research Center at the University of Michigan and the MICDE Fellowship offered by the Michigan Institute for Computational Discovery and Engineering. Lastly, I would like to extend my heartfelt thanks to my family members for their unconditional love and unwavering encouragement.

TABLE OF CONTENTS

Dedication . . . . . **ii**

Acknowledgments . . . . . **iii**

List of Figures . . . . . **viii**

List of Tables . . . . . **xiv**

Abstract . . . . . **xvi**

Chapter

1 Introduction **1**

1.1 Motivations 1
1.2 Dissertation Outline 6
1.2.1 Learning 7
1.2.2 Interpretation 8
1.2.3 Adaptation 10

2 Impact of Heterogeneity and Risk Aversion on Task Allocation in Multi-Agent Teams **12**

2.1 Introduction 12
2.2 Methods 15
2.2.1 Problem Formulation 16
2.2.2 Decentralized Deep Q-Learning with Beliefs 18
2.3 Results 23
2.3.1 Effect of Heterogeneity on Team Performance 23
2.3.2 Effect of Risk Aversion on Team Performance 26
2.4 Conclusions 30

3 Task Allocation with Load Management in Multi-Agent Teams **32**

3.1 Introduction 32
3.2 Methods 34
3.2.1 Problem Formulation 34
3.2.2 Embedding Agent Preference in Reward Shaping 36
3.2.3 Decentralized Deep Q-Learning for Load Management 38
3.3 Results 40
3.3.1 Effect of Idle Incentive and Reassignment Penalty on Load Management 42
3.3.2 Agent Importance 45

3.4 Conclusions . . . 48

4 Human-Autonomy Teaming in Immersive Environments . . . 50

4.1 Introduction . . . 50
4.1.1 Human-Autonomy Teaming . . . 52
4.1.2 Cognitive Workload . . . 53
4.1.3 Adaptive User Interface . . . 54
4.2 Methods . . . 54
4.2.1 Human-Autonomy Teaming Architecture . . . 54
4.2.2 Interfacing for Human Agents . . . 59
4.3 Experiment and Results . . . 64
4.3.1 Experiment Setup . . . 64
4.4 Human Subject Study . . . 68
4.4.1 Operation Scenarios . . . 69
4.4.2 Data Collection . . . 70
4.4.3 Exploratory Analysis Results . . . 72
4.5 Conclusions . . . 73

5 Evaluating Emergent Coordination in Multi-Agent Task Allocation through Causal Inference and Sub-Team Identification . . . 75

5.1 Introduction . . . 75
5.2 Methods . . . 78
5.2.1 Preliminaries . . . 79
5.2.2 Coordination in Different Stress Levels . . . 81
5.2.3 Sub-Team Identification . . . 82
5.2.4 Team Coordination . . . 83
5.3 Results . . . 84
5.3.1 Decision Strategies . . . 86
5.3.2 Coordination and Sub-Teams in Rule-based Strategies . . . 86
5.3.3 Coordination and Sub-Teams in Learned Strategy . . . 88
5.3.4 Underestimation and Overestimation . . . 91
5.4 Conclusions . . . 93
5.5 Algorithmic Recourse — Application of Reinforcement Learning in Explainable AI . . . 94

6 Multiagent Inverse Reinforcement Learning via Theory of Mind Reasoning . . . 97

6.1 Introduction . . . 97
6.1.1 Reinforcement Learning . . . 102
6.1.2 Inverse Reinforcement Learning . . . 102
6.1.3 Multiagent Inverse Reinforcement Learning . . . 103
6.1.4 PsychSim . . . 103
6.2 Methods . . . 104
6.2.1 Model Inference . . . 105
6.2.2 Decentralized MIRL . . . 105
6.3 Results . . . 107

6.3.1 Experiment . . . 108
6.3.2 Collecting and Visualizing Trajectories . . . 110
6.3.3 Model Inference Results . . . 111
6.3.4 Decentralized MIRL Results . . . 113
6.3.5 Unknown Teammates Setting . . . 116
6.4 Conclusions . . . 116

7 Bayesian Online Strategy Adaptation for Multi-Agent Teams in Open-World Novelty 119

7.1 Introduction . . . 119
7.2 Methods . . . 123
7.2.1 Preliminaries . . . 123
7.2.2 Offline Strategy Collection . . . 123
7.2.3 Bayesian Online Strategy Adaptation . . . 124
7.3 Results . . . 126
7.3.1 Convergence of Strategy Adaptation . . . 128
7.3.2 Novelty: Sudden Capability Degradation . . . 129
7.3.3 Impact of Team Size on Strategy Adaptation . . . 131
7.4 Conclusions . . . 132

8 Multi-Agent Team Co-Design of Behavior and Composition for Operational Requirements . . . 134

8.1 Introduction . . . 134
8.1.1 Composition Design and Behavior Design . . . 138
8.1.2 Multi-Agent Co-Design . . . 139
8.2 Methods . . . 140
8.2.1 Multi-Agent Task Allocation . . . 140
8.2.2 Multi-Agent Team Co-Design . . . 143
8.2.3 Iterative and Evolutionary Co-Design . . . 145
8.3 Experiment and Results . . . 150
8.3.1 Multi-Objective Rewards . . . 152
8.3.2 Iterative Design and Learning (IDEAL) . . . 154
8.3.3 Vote-Informed Co-Evolution (VICE) . . . 159
8.4 Conclusions . . . 161

9 Dynamic Task Planning for Smart Manufacturing — Applications in Reconfigurable Manufacturing Systems . . . 163

9.1 Dynamic Task Planning For Autonomous Reconfigurable Manufacturing Systems by Knowledge-based Multi-Agent Reinforcement Learning . . . 163
9.1.1 Introduction . . . 164
9.1.2 Methods . . . 165
9.1.3 Results . . . 169
9.1.4 Conclusions . . . 175
9.2 Intelligent Strategy for Learning and Teaming in Manufacturing Systems with Collaborative Autonomous Modular Machines . . . 176
9.2.1 Introduction . . . 176

9.2.2 Methods . . . 180
9.2.3 Experiment and Results . . . 183
9.2.4 Conclusions . . . 187

10 Conclusions, Summary of Contributions, and Future Work . . . 189

10.1 Conclusions . . . 189
10.2 Summary of Contributions . . . 193
10.3 Future Work . . . 195

Appendix . . . 198

Bibliography . . . 230

LIST OF FIGURES

FIGURE

2.1 Schematic of decentralized reinforcement learning approach in heterogeneous multi-agent task allocation problems 15

2.2 (a) Belief update mechanism for agents. For an agent assigned to task 1, the state estimator (SE) updates the belief $b(s_1)$, information estimator (IE) processes communicated observations and updates the belief over other tasks $b(s_j)$. The agent follows the policy π based on the overall belief updated for the next action. Comm Channel represents a communication strategy that can range from all-to-all to no communication. (b) Demonstration of state transition by time steps in team view. Allocation decisions for the next time step a^{t+1} are made based on agents' observations z and communicated information m received after the previous action a^t 17

2.3 Schematic of a Q-Network with an agent's belief over severity levels of 3 tasks as input and Q-Values of 2 actions as output. In the proposed approach, the agent's belief over task severity levels are used as an input to the Q-Network and the outputs are the Q-Values in the action space. 21

2.4 Learning curves with shaded area indicating standard deviations of every 20 training episodes and performance evaluations with 2000 runs for different team compositions with (a) heterogeneity in capabilities only (b) heterogeneity in capabilities and decision models 22

2.5 (a) Effect of number of noisy decision-makers in a team on the team performance, (b) effect of decision-making rationality on the team performance. Each point represents the mean value of 1000 evaluations. 27

2.6 Effect of belief reward threshold (b_{th}) on team performance, i.e., cumulative reward (top row) and first completion step of all tasks (bottom row). Results are compared to cases when the system provides a reward on full completion (full) of the tasks. The mean value and their corresponding standard deviations are obtained from 1000 simulations for each case. 28

2.7 Low (left), medium (center), and high (right) levels of added uniform prior in training. Blue lines indicate the average agent belief reward and green lines indicate the corresponding reward system would have given. Solid lines and shaded areas represent the mean value and standard deviations of every 20 episodes during the training. 28

2.8 Low (left), medium (center), and high (right) levels of added uniform prior in performance evaluation with a sudden event. 29

3.1 Schematic of a Q-Network with an agent's belief over severity levels of 2 tasks and previous assignment as input and Q-Values of 2 actions as output. In the proposed approach, the agent's knowledge state $\mathcal{K}$ is used as an input to the Q-Network, and the outputs are the corresponding Q-Values $Q(\mathcal{K}, \cdot)$ in the action space. 39
3.2 Effect of idle incentive R_{II} on operation effectiveness and capability usage with (a) no reassignment (b) medium reassignment and (c) high reassignment penalty T_{RP}. The performance evaluation includes 500 realizations, and each bar plot shows the mean and the standard error. 41
3.3 Effect of reassignment penalty T_{RP} on operation effectiveness (a) and load conservation (b), (c). Bar plots show the mean and the standard error for each performance metric evaluated using 500 realizations. 43
3.4 Demonstration of task allocation in an operation, where agents are trained (a) with no R_{II} and no T_{RP}, and (b) with R_{II} and medium T_{RP}. Black lines indicate agent decisions for the first 15 steps, and color blocks represent severity levels for each task ranging from 0 to 4. 44
3.5 Operation effectiveness comparison between having all agents (blue) and inactivating the least important agent (red) for (a) Heterogeneous Team, (b) Semi-Heterogeneous Team, (c) Homogeneous Team. Each histogram shows 200 realizations with the mean completion step indicated. 47

4.1 Methods of multi-agent task allocation in literature by team composition and task dynamics. 53
4.2 A game-based platform for humans to interact with autonomous agents in an immersive environment using virtual reality. Humans access information from both the environment and the virtual user interface. 55
4.3 Decentralized Partially Observable Markov Decision Process. At time step t, each agent makes a decision and executes the action. Then, the environment transitions to the next state with uncertainties. Agents receive rewards and local observation and communicate observed task levels with each other to make decisions in the next. . . . 57
4.4 Deep Q-network with load management. 58
4.5 PPG signal is recorded during a decision interval of 15 seconds, filtered (a), and converted to BPM (b). 60
4.6 Demonstration of the local environment (left) and the virtual adaptive user interface (right) accessible to the human agent. AUI Center – site information; top left – current site location; top bottom – decision count-down; yellow boxes – selection area for locations, tasks, and observations; blue box – suggested decisions. 61
4.7 Three interface choices providing either low, medium, or high information complexity level (left to right). 62
4.8 Training and operation of cognitive task load index adaptive user interface (CTLX-AUI) by estimating NASA TLX score. Red lines indicate the data is only used for training the interface. 63
4.9 Simplified NASA TLX Questionnaire. 64
4.10 Training and operation of cognitive task load adaptive user interface (CTL-AUI) by direct classification. Red lines indicate the data is only used for training the interface. . 65

4.11 The behaviors of a team of six autonomous agents (top 6) and one artificial human agent (bottom) on a 5-site fire and rescue operations. The team performance is evaluated with and without AHA to highlight the impact of human decision-makers. 66
4.12 Participant responses to pre- and post- questionnaires. (a) Their past experience with VR and game environments. (b) Their opinions on how comfortable working with AI teammates. (c) Their attention to AI teammates and decision processes. 72
4.13 Participant performance (steps to complete all tasks) in operation scenarios. (a) Without the sudden event. (b) With the sudden event, compared to the all-autonomy team. . 73

5.1 Demonstration of aggregating short decision time series in $N = 3$ stress levels across E simulated episodes, for an operation with $p = 6$ tasks and $l = 4$ maximum task demand level. 80
5.2 A weighted directed network of a team of five agents with edges represented by behavior prediction skills. Solid orange arrows indicate strong predictability given an agent, dashed blue bidirectional arrows indicate weak predictability between agents, and shades indicate the two sub-teams. 82
5.3 (Panels A.1-4) The coordination for 4 stress levels for a team of 30 homogeneous agents in an operation with 50 homogeneous tasks, following cooperation and patrolling strategies. The coordination score $\rho \in [0, 1]$ between two agents is measured by TA-CCM with 1000 episodes. (Panels B.1-2) High and low correlations between observed and predicted decisions of agent 3 given agent 1. 85
5.4 Sub-team identification for three 10-agent teams under cooperation rules or RL trained policy in 3 stress levels. Solid yellow boxes indicate identified sub-teams and dashed yellow boxes show ungrouped agents. 88
5.5 Coordination scores for three teams at different stress levels. 89
5.6 Sub-team identification and coordination scores for the 10-agent homogeneous team under RL-trained policy, using (a) Pearson correlation and (b) mutual information. . . 91
5.7 Performance of the team of 10 RL-trained homogeneous agents when making rational decisions, compared to overestimating and underestimating the stress level. 92
5.8 Reinforcement learning formulation for algorithmic recourse. 95

6.1 Overview of the proposed approach for Multiagent Inverse Reinforcement Learning via Theory of Mind Reasoning (MIRL-ToM). Given the environment and the demonstrated behaviors of a team of humans, each learner agent performs single-agent IRL by reasoning about other agents' behavior via ToM and output the resulting reward functions. 98
6.2 Two phases of the Multiagent Inverse Reinforcement Learning via Theory of Mind. . . 104
6.3 Forward pass of feature count estimation. Each learner agent uses the model distribution data to select the actions of other agents and select an action given the current reward model. The feature counts are computed from the environment states and the team actions. 107
6.4 Transition dynamics of the victim status. $\boldsymbol{search}$ either finds a victim or an empty cell, $\boldsymbol{triage}$ prepares a victim, and it requires two agents at the same location to $\boldsymbol{evacuate}$ it. 108

6.5 An example trajectory at timestep $t = 4$ (left), $t = 10$ (center), and $t = 16$ (right) for the Medic (top) and Explorer (bottom) agents. 110
6.6 Average model probabilities with standard deviations (shaded) of 16 trajectories for Medic (top) and Explorer (bottom) agents under two experimental conditions: ***cond***-3 (Gt-Op-Rd, left) and ***cond***-4 (Rd-Tk-Sc, right). 111
6.7 Learning curves of the reward weights during IRL for Medic (top) and Explorer (bottom) in two experimental conditions: ***cond***-2 (left) and ***cond***-3 (right). 112

7.1 Overview of BOSA in addressing open-world novelty. (A) Successful completion in the same operation as training. (B) Failure when the capability degradation novelty is introduced. BOSA with a Bayesian strategy selection mechanism can achieve fast adaptation in novelty. 120
7.2 Applying transfer learning for Q-networks to obtain strategies under novel situations . 124
7.3 Bayesian online strategy adaptation mechanism 125
7.4 Strategy belief convergence under 4 conditions with different initial belief distributions. For each condition, we run 100 realizations and evaluate the step of completion and success rate within 30 steps. The realizations are then separated based on completion steps. Top plots show the cases that the team completes the operation with at most 9 steps; bottom plots show longer-step or failing cases. The shaded area indicates the standard deviation of the strategy belief. 127
7.5 Strategy adaptation under a sudden capability degradation at step 3 with two strategy selection heuristics: (a) Greedy and (b) Inclination. Black lines indicate agent decisions over steps. Blue colors indicate the remaining task demand level for each location. The number on top of the colored cell is the total capability applied, summed over the agents selecting the same task. 130

8.1 Multi-agent team co-design for operational requirements 135
8.2 Crossover and mutation phases in conventional Genetic Algorithms with an analogy in attributes of agents. 143
8.3 A co-design framework by Iteration Design And Learning (IDEAL). Starting at a given team composition, module (a) *Behavior Learning* performs multi-agent reinforcement learning with centralized information on tasks and capabilities applied, individual rewards, and decentralized decision models. With an evaluation procedure and performance metrics on learned behavior, module (b) *Composition Design* determines whether to remove or add agents based on constraints. With the adjusted team composition, the iterative process continues until the drop performance or no change in team composition. 144
8.4 Two cases in the migration phase for Vote-Informed Co-Evolution (VICE) to attract the best agents from other teams to the best team with the voting mechanism: (a) when an equivalent agent exists in the best team, up-vote and (b) otherwise, generate an offspring for the best team. 148
8.5 An illustration of a disaster-relief operation. The team of agents equipped with task-specific attributes described by capabilities decides which task to execute under dynamic task demands and uncertainties. 151

8.6 Training curves of average agent reward for the heterogeneous team composition with multi-objective reward weight ω_r ranging from 0 to 3.0. Shaded colors indicate the moving standard deviations of the training rewards in the previous 30 episodes. 153
8.7 The behavior of the heterogeneous team in (a) an optimistic and (b) a pessimistic case under uncertainties. Black lines are agent decisions over steps, blue cells show the remaining task demand levels, and the red numbers indicate the capabilities applied. . 153
8.8 Team selection by IDEAL. For each iteration, we show (top) the median of team capability usage for each attribute (U_{ext}, U_{res}) and the distribution of the total capability usage U_{tot}, (middle) the distribution of the number of steps to complete tasks, and (bottom) an episode of the team behavior after iteration 6. 155
8.9 Team adjustment by IDEAL starting from the initial 5 heterogeneous agents from Sec. 8.2.2.1. (a) The team receives penalties on total remaining demands $R_D = -\sum_j s_D^j$, and the final team behavior is shown after iteration 3. (b) The team receives modified penalties $R_D = \min(-4, -\sum_j s_D^j)$, and the final team behavior is shown after iteration 4. The top plots show the distributions of capability usage and completion efficiency per iteration for both design processes. 157
8.10 Performance of VICE (blue) compared to ASCE (orange) and multi-agent genetic algorithm without votes (green) in different problem complexities. Team Score Sum tracks whether all 3 teams find at least 2 optimal team compositions. The star and triangle symbols indicate when the first and second solution types are found, with the corresponding iteration numbers in color shown near axes. 160

9.1 Digital twins in reconfiguration manufacturing systems. 164
9.2 Decentralized deep Q-learning for the KDP framework. 169
9.3 Example parts with different features, adapted from [1]. 170
9.4 a) A production scenario for two specialized machines; Task plan visualizations in b) an optimistic case and c) a pessimistic case. 171
9.5 A production scenario with 4 reconfigurable machines. 172
9.6 Training (a) and performance measures (b)(c) for different levels of penalties on reconfiguration and module usage. 173
9.7 Strategy visualizations for 4 reconfigurable machines, with a) team decisions over 10 steps and b) strategy snapshots at step 3 and 5. 174
9.8 Components of Intelligent-Anarchic Learning paradigm. 179
9.9 Flexible and collaborative decision-making in I-AL. 180
9.10 Design-while-Learn framework has an iterative process of (a) I-AL using deep reinforcement learning and (b) heuristic-based modular design. 183
9.11 Task requirements in a shop floor operation. 185
9.12 (a) Comparison of training curves and (b) strategy visualization of a team using MOR (blue) and SOR (red). Colors indicate task demands and stages, numbers indicate assigned modules, and lines represent decisions. 186
9.13 Team member selection process for each DwL iteration shown with (a) team size and the selected team, (b) module usage for both types, and (c) evaluated performance over 1000 realizations. 187
9.14 Team adjustment process using DwL, starting with the initial 4 agents (top) and visualizing the final team strategy. 187

A.1 Recourse policies for loan approvals. Policy A has only one feature change (low sparsity) but with a high failure rate; Policy B has the lowest expected cost but might result in a situation that costs more to recover; Policy C has a slightly higher expected cost than Policy B but lower variance in cost (risk), which can be considered as a safer Policy. 198
A.2 Visualizing risks in three recourse policies from Figure A.1. The x-axis indicates the cost, and the line (outcome) thickness indicates the outcome probability. 201
A.3 Policy visualization Example in German Credit. 214
A.4 Health insurance premiums recourse (synthetic) 226
A.5 Moving average of total recourse cost using G-RSEVI with maximum episode 10,000 and exponential decay factor 0.9995 for ϵ. Shades indicate $\pm\sigma$ cost range of the moving window. 228
A.6 Policy visualizations; $\beta = 0, 0.5, 1.0$ correspond to risk-neutral, medium risk-averse, and high-risk-averse respectively. 229

LIST OF TABLES

TABLE

3.1 Two-Location Traversing Cost Matrix 37
3.2 Capabilities for a Heterogeneous Team 42
3.3 Agent Importance for Three Teams 46

4.1 Test Accuracy of CTL-AUI Variants 67

5.1 Team Specifications 90

6.1 Ground-truth reward functions for the two agents as linear combinations of features ϕ_i weighted by $\boldsymbol{\theta}_i$. 109
6.2 Baseline reward profiles 111
6.3 Learned reward functions for the two agents, for all conditions in the known teammates setting. 113
6.4 Similarity of recovered behavior for the two agents, for all conditions in the known teammates setting. 115
6.5 Similarity of recovered behavior for the two agents in the unknown teammates setting. 117

7.1 Collected task improvement likelihoods for 2 fire tasks using three strategies at different fire demand states. 128
7.2 Comparison of BOSA strategy selection heuristics and noisy rational decision-making mechanisms. 131
7.3 Evaluating the effect of operation size on strategy adaptation under the same level of capability degradation. N_{degrad} is the number of degraded cases being evaluated, with N_{total} indicating the number of all possible cases. ρ_{half}, ρ_{full} measure the percentage of cases that complete half of, all of the tasks respectively. 132

8.1 Notations Used in Performance Evaluation 146
8.2 Performance Evaluation for Two Team Compositions 152
8.3 Parameters of Team Selection Co-Design Iterations 156
8.4 Team Adjustment Co-Design Iterations 158
8.5 Final Team Compositions and Requirement Satisfaction 158

9.1 ModDec-POMDP formulation 181
9.2 Adaptive modular design (AMD) heuristics 184

A.1 Evaluating recourse policies with different risk-aversion levels β and horizon $H = 12$ for AID, GCD, and two example instances sampled from the datasets. n denotes the number of instances used for evaluation. For each risk-aversion level, we compute a recourse policy and apply it to each instance (initially with the undesired outcome) in the dataset. Due to the uncertainties in action outcomes, we estimate the average of each measure using simulated 100 trials. Then, the average across all instances for each metric is computed. The best metric values among the policies are highlighted in bold. 210
A.2 Dataset Overview 211
A.3 Evaluating recourse policies for gender for Adult Income and German Credit datasets; the same evaluation procedures are followed as Table A.1. The male group is exposed to less risk under the policy with the same risk-aversion level. 211
A.4 The feature list and discrete values for both datasets. Bolded features are immutable. The other features are mutable, and Age is non-actionable. 216
A.5 Action Cost and Transition Models; If the action fails, the state stays the same. 217
A.6 Action Cost and Transition Models; If the action fails, the state stays the same. 219
A.7 The feature list and discrete values for HIPD. Bolded features are immutable. The other features are mutable and actionable. 221
A.8 Action Cost and Transition Models; If the action fails, the state stays the same. 221
A.9 Compare risk-aware policies to the policy computed by using the final CEs provided by DiCE; We randomly select $n = 100$ individual instances from AID and GCD; the same evaluation procedures are followed as Table A.1. 222
A.10 Mann-Whitney U test results for cost-mean and cost-variance between gender groups for AID and GCD 222
A.11 Mann-Whitney U test results for VaR_{80} and CVaR_{80} between gender groups for AID and GCD 223
A.12 Mann-Whitney U test results for VaR_{95} and CVaR_{95} between gender groups for AID and GCD 223
A.13 Evaluating recourse policies across gender using a different set of cost and transition functions; the same evaluation procedures are followed as Table A.1. 224
A.14 Parametric evaluation on risk-aversion β and horizon H for AID and GCD; the same evaluation procedures are followed as Table A.1. The time spent to compute the policies for all states (57600 in AID, 147456 in GCD) is included. 225
A.15 Evaluating recourse policies for Health Insurance Premiums Dataset; the same evaluation procedures are followed as Table A.1. 225
A.16 Evaluating recourse policies by penalizing LPSD with different risk-aversion levels β and horizon $H = 12$ for AID, GCD; the same evaluation procedures are followed as Table A.1. 225
A.17 Evaluating risk measures 226

ABSTRACT

Coordination strategies in multi-agent systems are transforming real-world applications, enabled by intelligent decision-making and dynamic synchronization among heterogeneous entities. Human-autonomy teaming (HAT), leveraging the diverse capabilities of autonomous agents and humans, can achieve superior performance than either could accomplish alone across various domains. As autonomous agents are increasingly thought of as teammates alongside humans in large-scale multi-agent task allocation (MATA) operations, they share equal responsibilities in situational perception, strategic planning, and task execution, instead of replacing humans or being subordinate members. Such decentralization of decision-making and coordination can enhance efficiency, scalability, and flexibility in HAT. However, the inclusion of humans within autonomous systems requires a significant effort to prepare autonomy for strategically collaborating with humans in dynamic, uncertain, and adversarial environments and for adapting to operating envelopes beyond the learned expertise. The coordination strategies currently lack considerations of collaborative task planning, load management, behavior explanation, and adaptation in unexpected situations. Effective synergy between humans and autonomous agents necessitates the development of risk-aware, interpretable, and adaptive coordination strategies in dynamic MATA operations.

To address the challenges of developing AI-powered autonomous teammates, this dissertation presents methods in three phases: 1) enabling autonomy to learn strategies for coordinating with human teammates and achieving operation objectives, 2) empowering autonomy to interpret teaming dynamics and explain preferences of teammates, and 3) equipping autonomy with adaptability in collaborating with unknown teammates and responding to evolving and novel situations. First, a decentralized heterogeneous teaming decision-making framework is developed to maximize the effectiveness of MATA operations by incorporating human characteristics and explicitly modeling the dynamics between agent capabilities and task demands. The framework is enhanced by

enabling idling behavior and avoiding overload through multi-objective reward shaping that considers completion efficiency and load reduction. A game-based platform is then developed to evaluate HAT in disaster relief operations, where human decision-makers, interfaced with a workload adaptive user interface providing suitable information complexity and decision recommendations, interact with autonomous teammates to effectively accomplish the mission together. With learned behavior, interpretation can facilitate understanding how certain decisions are made. An interpretation technique is proposed to uncover emergent behavior by quantifying team coordination using causal relationships among agent decision patterns and by identifying highly synchronized sub-teams through a representation of weighted networks. To explain the behavior of initially unknown teammates as decision support, a Multi-agent Inverse Reinforcement Learning method incorporating Theory of Mind reasoning is proposed to infer the preferences of teammates for robust collaboration by observing behavior during operations. Adaptation cannot be achieved without the capability of learning and interpreting. A Bayesian Online Strategy Adaptation algorithm is introduced to adapt instantly when facing unexpected events in urgent operations, by preparing a strategy pool offline, observing unusual dynamics caused by novelty, and inferring the most suitable strategy. The effectiveness is validated in a novel capability degradation event. Lastly, to seek superior operational performance, simultaneous evolution of team composition and behavior can automate the design cycle of MAS. A multi-agent team co-design framework is proposed by understanding the interdependence between tasks, team composition, and coordination strategies. An iterative approach, alternating between behavior learning by multi-agent reinforcement learning and heuristics-based composition design, demonstrates the process of designing teams for MATA operations and provides a cost-effective analysis of diverse design options.

CHAPTER 1

Introduction

1.1 Motivations

Recent developments in autonomy and multi-agent systems (MAS) offer opportunities that might lead to a paradigm shift in real-world applications including emergency response [2], transportation management [3], smart manufacturing [4], healthcare system organization [5], and conversational assistance in business [6]. When operations require multiple entities distributed dynamically, functionally, and spatially to address heterogeneous, time-extended, and cross-dependent tasks, human-autonomy teaming (HAT) [7] can achieve superior effectiveness in completing tasks that cannot be accomplished individually.

With the advancement in artificial intelligence (AI) and the increase in computational capacities, autonomy is transforming the role from a subordinate member to a teammate who shares equal responsibilities as a human agent in perceiving, planning, and execution. Such decentralization in decision-making can improve the performance, flexibility, resilience, and scalability of human-autonomy teams when handling large-scale operations or encountering unexpected events. The inclusion of humans in autonomous systems poses challenges in aspects of collaboration, strategic task planning, load management, behavior interpretation, and adaptation to novelties. To ensure the coordination and synchronization between humans and autonomy, there exists a pressing need to develop methods that provide effective, risk-aware, explainable, and adaptive coordination strategies in dynamic task allocation operations for human-autonomy teams. Therefore, the goal of this research is to prepare AI-powered autonomous teammates with heterogeneous capa-

bilities to seamlessly collaborate with humans in multi-agent task allocation (MATA) operations. This requires the development of coordination strategies in three phases: learning to collaborate and achieve objectives, interpreting learned decision behavior and preferences of teammates, and adapting strategies to unknown teammates and unexpected events.

In the first phase, learning coordination strategies for autonomous teammates in MATA operations involves both task assignment and scheduling with multi-level subtasks and conditional dependencies on prerequisite tasks and applied capabilities of team members, which belong to the complexity category [ST-MR-TA] (complex dependencies, single-task agents, multi-agent tasks, time-extended assignment) according to the complexity definition of taxonomy [8]. To date, most work for MATA operations has focused on homogeneous team compositions [9–11] and heterogeneous teams with specialized capabilities for handling various tasks. The solution approaches include mixed-integer programming [12–16], hierarchical planning [17, 18], or genetic algorithms [19]. To equip agents with their own decision-making processes, decentralized partially observable Markov decision processes (Dec-POMDPs) [20] are used to formulate MATA operations, allowing agents to gain intelligence by multi-agent reinforcement learning (MARL) [21–23] through the interaction with operation environments and enabling autonomy to work alongside humans as team members in more complex task environments [24].

In HAT, autonomous teammates are capable of handling dangerous tasks but are limited in reaction to unforeseen events. Humans have more adaptive and creative problem-solving skills but are limited in handling cognitive workload [25, 26]. Although coordination algorithms [22, 27, 28] allow the team to accomplish tasks efficiently, there is a lack of risk considerations, where additional strategic amendments are required under unanticipated situations. To mitigate such risks, team members should be intelligent enough to prepare for the unknown by avoiding overload and staying unoccupied [29, 30]. Being constantly active would be especially harmful to human agents and cost extra resources. There is limited research on exploring the effect of having idle action as an option and considering load management and capability usage as secondary operation objectives. In addition to explicitly incorporating the dependency between heterogeneous task-specific

capabilities of agents and task dynamics, the difference in risk tolerance during decision-making also plays a significant role in overall teaming performance, as humans might take actions that are considered sub-optimal, risky, surprising, or contrasting the previous experience in response to unexpected events [31–33]. Therefore, by taking advantage of the scenario reproducibility, the process of modeling, training, evaluating autonomy in simulations, and interfacing with actual human teammates before deployment serves as a critical step in the development of human-autonomy teams.

Despite the capability of learning teaming behavior, it often remains unclear how decisions are made by autonomous teammates. In the second phase, interpreting the learned strategies can provide an explanation of how coordination has emerged and offer an understanding of how the roles of teammates have evolved. In the Dec-POMDP formulation for MATA operations, coordination strategies are naturally formed and driven by a common goal without explicit communication of intended actions. Understanding complex interactions, behavior causality, and emergent strategies of MAS remains challenging. To address this issue of black box strategies, explainable reinforcement learning [34–36] aims to create interpretable reward functions for evaluating the interestingness elements in agent behaviors such as frequency, execution uncertainty, and favorable/adverse situations [37]. However, these approaches have not explained the correlation or causality effect between agents, and they are not general enough to merely use state and decision information. The Pearson correlation [38] and KL divergence [39] are widely used to measure the relationship between two random variables, to match the probability of the desired strategy [40], or to identify strategy switching [41], but these metrics do not address the fact that correlation does not imply causation. In economics and ecosystems, statistical tests including Granger Causality [42] and Convergence Cross Mapping (CCM) [43] have been developed to detect the causality between time series of two variables. While Granger Causality assumes that variables are separable, CCM variants [44–47] seek synergistic effects between variables in stochastic and dynamical systems. Causality has been rarely studied in MATA to quantitatively measure the level of coordination in strategies. Although correlation in rewards [48], performance metrics [49], and behavior visual-

ization [50, 51] help illustrate the evolving behaviors during learning, the coordination patterns depicted by agent decisions have not been studied, and whether the suitable strategy is implemented facing unexpected situations is therefore not guaranteed. There is a need to quantify the synchronization and synergy of teaming behaviors under different operations to gain insights into the underlying team dynamics and the emergent coordination strategies.

The insights gained from interpreting the strategies humans use in collaborative teaming can be further leveraged as decision support for MAS to predict team behavior, understand collaboration dynamics, and improve performance. This is particularly difficult under uncertain situations, where individuals collaborate for the first time and lack knowledge of each other's goals and strategies. A robust method to understand individual behavior in collaborative tasks is to assume that it results from a reinforcement learning (RL) process, where a reward function encodes preferences. By recovering this reward function from behavior traces, the underlying strategy of a teammate can be explained and understood. This approach is known as Inverse Reinforcement Learning (IRL) [52, 53]. Understanding the strategies and intentions underlying the behavior of a team in collaborative tasks requires extending IRL to multi-agent settings, known as Multi-agent Inverse Reinforcement Learning (MIRL). Generalizing IRL to MAS requires the inference of the individual reward functions guiding the behavior of each agent and explaining how each agent behaves and how together the agents coordinate, given trajectories of team behavior. Within MIRL, it is often assumed that the team behavior is the result of an equilibrium solution concept, such as Nash equilibrium [54, 55]. However, challenges arise as multiple equilibrium solutions exist given a combination of individual rewards. The methods that find the equilibrium [55–58] still assume that agents have perfect knowledge of the goals and intentions of their teammates, ignoring the dynamic nature of interactions and the possibility of first encounters. Enabling the capability of Theory of Mind (ToM) [59] in MAS becomes essential, where agents achieve robust collaboration by ascribing mental states to explain and predict the actions of others.

Being capable of interpreting team behavior and operational situations allows the development of adaptive coordination strategies in the third phase. Adaptability enables autonomous teammates

to operate in environments beyond the learned expertise. MARL approaches [27, 60–62] provide MAS with coordination strategies to effectively operate in dynamic yet enclosed environments, but the learned expertise cannot be guaranteed in the presence of the unknowns and does not generalize to open-world novelty [63], i.e., unexpected scenarios diverging from the known properties of an environment. Adaptation is particularly crucial for MAS in collaborative tasks, as even minor disturbances can cascade into failures. IRL methods [53] learn the intrinsic reward functions from expert demonstrations as the ability of adaptation and have shown promising results in multi-agent applications such as adapting behavior to unknown teammates [64], adapting a single educator to various learners [65], and adapting driving behavior to pedestrians or unfamiliar environments [66]. However, the learned reward functions do not directly identify the characteristics of novelty in the open world, and an out-of-distribution novelty could completely fail the system. With the increasing need for generalizing capabilities in novel situations, research has also focused on the paradigm of open-world learning [67] —agents with limited experience revise their expertise to perform beyond known conditions. Methods shift toward online adaptation and continual learning [68, 69] for single-agent problem domains, where agents can adjust the learned decision models based on recent experiences [70] or global feedback [71]. Another category of approaches attempts to infer novelty and update the decision-making process [72–75]. However, in a novel scenario demanding an urgent response, these methods would still suffer from slow adaption or require additional training. Methods that can achieve instant online adaptation under novelty by observing unusual dynamics without re-training are desired.

The team composed of heterogeneous but pre-defined capabilities often defines the upper limit of the performance in an operation. Adapting or evolving the team composition along with behavior learning could push this performance limit. In the conventional design cycle of a MAS, the configurations or the team composition are pre-determined, and the behavior is then optimized for certain tasks. When they do not perform as expected, they have to be re-configured, re-programmed, and re-evaluated [76]. This becomes a trial-and-error process and would require tremendous effort and investment. For MATA operations, the development focuses not only on

the individual agent configuration but also on the interaction and collaboration between them to achieve operational requirements, significantly escalating the complexity and challenges of the design process. There is a research gap in automating the MAS design process in simulations and generating diverse design options with a better understanding of the interdependence between operational requirements, task dynamics, agent capabilities, team composition, and strategic behavior. Behavior design in MATA operations refers to being capable of making decisions and taking actions given observations or knowledge of the tasks [9, 77]. Team composition design determines whether to include agents in the team given their utilities and performance metrics in an operation. Works have used optimization [78], RL [79], and evolutionary computing [80–83]. Due to the complexity of MAS, maintaining a population of teams becomes intractable. The concept of competitive and cooperative co-evolution [84, 85] is introduced that uses comparative information to modify team composition with applications in swarm intelligence [86]. However, these works assume that agent behavior does not change (e.g., fixed controllers and pre-defined rules) over generations, and the question of whether the team composition can reach optimal performance in an operation is not answered. Inspired by single-agent control co-design [87–92], approaches for MAS leveraging the evolution of team composition can drive team behavior to achieve optimal performance in operations. With complex interaction between agents, co-design for MAS is more challenging because a slight change in composition could lead to a completely different behavior. Co-design for heterogeneous teams also uses evolutionary computing [93] that puts the parameters of both team composition and agent controllers into evolution. However, there is limited work in multi-agent team co-design problems for MATA operations, through which agents simultaneously evolve task-specific attributes rather than formation and operational strategic reasoning capability instead of control actions.

1.2 Dissertation Outline

Motivated by an array of challenges presented in current MAS and given the increasing need for synergistic coordination between humans and autonomous agents for complex MATA operations,

this research aims to transform the development of multi-agent teams and introduce novel methods that enable AI-powered autonomous teammates to 1) *learn* coordination strategies with human teammates, 2) *interpret* teaming behavior to uncover the underlying dynamics among teammates and operation environments, and 3) *adapt* strategies even team composition to maneuver under uncertainties, react to novelties, and seek optimal operational performance. The summary of the contributions toward these goals is provided in Sections 1.2.1, 1.2.2, and 1.2.3 respectively.

The chapters of this dissertation are compiled from a series of papers that have been published, accepted, or submitted to academic conferences and scientific journals. Certain background, preliminaries, and formulations may reiterate previously covered information

1.2.1 Learning

This phase first presents the development pipeline including modeling, training, and validating coordination strategies, enabling autonomous teammates to collaborate with actual human decision-makers to achieve operation objectives together.

In Chapter 2, an AI decision-making framework for multi-agent heterogeneous teams is developed to learn dynamic task allocation decisions and maximize the operation performance through reinforcement learning. The heterogeneity is described by 1) the difference in agent capabilities of task handling, sensing, and communication that have a direct impact on task demand dynamics and perceived information accuracy and 2) the difference in agent decision-making process representing the level of risk aversion. The framework extends Dec-POMDP [94] to the heterogeneous teaming where agents are equally responsible for strategic planning and execution, and the heterogeneity in agent capabilities and decision processes are explicitly modeled. The proposed approach employs deep learning techniques [95] to improve computational efficiency and utilizes belief representation to summarize past experience, allowing the incorporation of risk tolerance in the framework. Leveraging this framework, some characteristics of humans are incorporated into the team, and a thorough analysis of the effect of heterogeneity and human characteristics on teaming performance is conducted in this chapter as a basis to facilitate the study of HAT.

Load management functionality in Chapter 3 is built upon the framework developed in Chapter 2, which addresses the concerns of human overload and resource over-usage. Multi-agent teams with reduced load respond more reliably in emergencies that require extra load/resources or in situations where one of the team members is compromised. In Chapter 3, the decision-making framework is further enhanced by considering the idling behavior as a secondary objective or an incentive to reduce load and save resources described by agent capability usage and task reassignment. The experiments show that, with a carefully designed reward mechanism, the optimal idling behavior has a negligible impact on completion efficiency but would avoid unnecessary capability usage and task reassignment, making the team more resilient in unexpected situations.

In Chapter 4, a game-based HAT platform is developed to quantitatively analyze the performance of HAT in a disaster relief scenario. This platform allows human decision-makers to interact and collaborate with autonomous agents in a high-fidelity immersive environment to accomplish the mission. Autonomous agents with heterogeneous capabilities are equipped with the AI decision-making processes proposed in the previous chapters and are trained to dynamically and effectively allocate tasks in response to operational demands. Humans can perceive the environment by assessing the local situation and receiving information through a virtual adaptive user interface (AUI). By monitoring human cognitive workload, the AUI is trained to provide a suitable level of information complexity to assist decision-making and situation awareness. With the HAT platform, a dataset consisting of the decision behaviors and cognitive workload measures of 50 human participants is collected to study the coordination and communication of HAT in reproducible time-critical operations. An exploratory analysis demonstrates the effectiveness of the inclusion of AUI and decision recommendations across various scenarios.

1.2.2 Interpretation

To facilitate the explainability of learned strategies and behavior understanding among agents, the second phase proposes techniques for autonomous teammates to interpret coordination strategies and infer roles or preferences of unknown teammates for better collaboration.

In Chapter 5, a quantification method is developed by utilizing the concepts of decision causality [43] and bidirectional weighted networks [96] to evaluate coordination in MATA operations. Visualizing and measuring whether agent decision patterns can be informed by past decisions of other agents and whether certain agents form sub-teams that are highly synchronized compared to others are crucial for understanding emergent strategies and recognizing the need for them in evolving situations and unforeseen events, especially when agents are facing extreme workload and stress, and when agents over- or under-estimate the task demand. The developed metrics quantify team coordination with predefined centralized rule-based strategies and decentralized team strategies trained by MARL and show that RL-trained teams can learn more highly coordinated behaviors, compared to rule-based policies. This method is the first attempt to apply causal inference for evaluating coordination behaviors in MATA operations, which is capable of quantitatively investigating agent-level coordination, sub-team formation, and team-level strategies. This quantification method serves as a post-hoc explanation tool for teaming behavior. Additionally, a generalized post-hoc explanation technique is introduced in Appendix A that connects the explainable AI literature with risk-sensitive reinforcement learning and leverages explanations to provide recommendations on how to sequentially change the model input (features) to reach a desired output with risk considerations.

To understand how humans interact with each other in collaborative settings, especially when individuals know little about their teammates, Chapter 6 introduces Multiagent Inverse Reinforcement Learning via Theory of Mind Reasoning (MIRL-ToM). It extends the current MIRL approaches that infer the reward functions guiding the team behavior given trajectories of team behavior and allows agents to interpret the goals of others based on perceived behavior observations instead of assuming that team members know each other's goals. For each agent, ToM reasoning estimates a posterior distribution over baseline reward profiles/roles given their demonstrated behavior. Then MIRL is performed via decentralized equilibrium by employing single-agent Maximum Entropy IRL [97] to infer an explainable reward function. During this process, the behavior of other teammates is simulated according to the time-varying distribution over profiles. MIRL-

ToM is evaluated in a simulated 2-player search-and-rescue operation where the goal of the agents, playing different roles, is to search for and evacuate victims in the environment, and results show that the choice of baseline profiles is paramount to the recovery of the ground-truth agent roles and that MIRL-ToM is able to recover the rewards used by different agent roles, interacting both with known and unknown teammates.

1.2.3 Adaptation

The third phase equips autonomous teammates with adaptability in handling unexpected situations using the learned expertise in understanding teaming behavior and inferring novel operational dynamics. As a step further toward the real-world deployment of MAS, where minor disturbances can cascade into catastrophic failures and immediate interventions are needed in unexpected situations, Chapter 7 focuses on fast adaptation by selecting the most suitable team strategy to deploy. The Bayesian Online Strategy Adaptation (BOSA) algorithm consisting of three phases is developed to address novelty in MAS: 1) In offline preparation, a pool of feasible strategies handling various situations and the corresponding likelihood of unusual event dynamics caused by novelty are collected. 2) During online inference, a probabilistic distribution (belief) over the collected strategies is maintained and updated to infer the most probable strategy to use based on online observations of unusual events. 3) Online adaptation is then achieved by strategy selection heuristics leveraging the changes in the belief. Experiments are then designed to validate the strategy inference mechanism, investigate how BOSA can scale for larger operations, and demonstrate BOSA can effectively address novel agent capability degradation events within one decision step with a strategy selection heuristic on belief inclination.

While Chapter 7 achieves the adaptation of team behavior in novel situations, another form of adaptation is the evolution of team composition to seek the upper limit of the operation performance. In Chapter 8, a multi-agent team co-design framework is proposed to simultaneously evolve team behavior and team composition with the goal of automating the design cycle of MAS in simulations to reach the superior operational optimum by understanding the interplay

between operational dynamics, team composition, and strategic teaming behavior across different design levels. An iterative co-design approach through team selection and team adjustment is presented, using MARL in Chapter 2, interpretable multi-objective behavior evaluation in Chapter 3, and heuristics-based composition design. Accounting for multiple operational optimums, a vote-informed co-evolutionary algorithm is introduced to seek diverse solutions to multi-agent operations that outperform the evolution based on individual fitness. The co-design experiments demonstrate the effectiveness and trade-offs of the approaches in disaster relief operations and multi-attribute operations. In addition, the application of the co-design framework in reconfigurable manufacturing systems is summarized in Chapter 9, where a team of autonomous modular agents can effectively collaborate, plan tasks, and switch modules for complex shop floor production operations.

CHAPTER 2

Impact of Heterogeneity and Risk Aversion on Task Allocation in Multi-Agent Teams

This chapter serves as the first step to enable autonomous agents to learn coordination strategies in MATA operations, considering the heterogeneity in task-handling capabilities, operation objectives, and decision-making rationales.

2.1 Introduction

Recent developments in autonomy offer opportunities that might lead to a paradigm shift in several domains. A multi-agent system design is beneficial in many aspects, particularly when a system is composed of multiple entities that are distributed functionally or spatially. Collaboration enables the agents to work as a team and complete activities that they are not able to accomplish individually. Instead of agents being centrally controlled, a decentralized multi-agent team can improve performance, robustness, and scalability by planning and performing actions in parallel.

To date, most work in multi-agent task allocation (MATA) [9] has focused on homogeneous team compositions [10] with the consideration of temporal constraints [11], communication protocols [21], and spatial dynamics of tasks [22]. Task allocation for heterogeneous teams [13–16] has also been addressed to study the collaborative behaviors of heterogeneous agents with specialized capabilities for handling various types of tasks. The solution approaches of MATA in literature include optimization-based methods [10, 11, 13–16] modeled as mixed integer programming problems and reinforcement learning (RL)-based methods [21, 22] which are formulated as

Decentralized Partially Observable Markov Decision Processes (Dec-POMDP) [20] and solved by deep learning techniques [95, 98, 99] or heuristic search [94]. As computing systems continue to advance, there is a push towards considering more complex environments and diverse teams such as autonomous agents as team members alongside humans in many real-world applications [24], ranging from transportation systems to exploration of hazardous environments and rescue in disaster scenarios. Autonomous agents are capable of handling dangerous tasks but are limited in reaction to unforeseen events. At the same time, humans have more adaptive and creative problem-solving skills but are limited in terms of handling specific tasks and cognitive loads. This inclusion of autonomy within a team context establishes a need to develop a framework to optimally train a team of heterogeneous agents and investigate how a team benefits from heterogeneity in an operation. The literature, however, lacks enhanced modeling of strategic task allocation for teams of heterogeneous agents that are managing dynamic task demands in urgent and uncertain operations. While heterogeneity in multi-agent teams may offer benefits, these benefits can only be understood by analyzing optimal heterogeneous team compositions and dynamically distributing tasks among agents in uncertain environments, which remain challenging. Although optimization-based MATA problems can handle constraints between heterogeneous tasks and agents [14], the task demands are usually satisfied upon one-step assignment [16] and analysis of dynamically changing demands affected by agent decisions and capabilities is still limited [13, 15]. In particular, a Dec-POMDP formulation has not been formally utilized by a team of heterogeneous agents to provide the comprehensive modeling of the dynamics between agent capabilities, task demands, and perception accuracy.

Unlike homogeneous teams where agents have the same functional capabilities and share the same parameters in their decision processes, heterogeneous team composition and training require a delicate effort considering the trade-off between the stability and adaptive behavior of each agent. Heterogeneity in a team is often considered as the difference in rules of engagement or assignment constraints [14], task suitability [15], or functional heterogeneity [16]. In addition to the heterogeneity in agent task-related capabilities, the difference in agent risk tolerance in decision-

making, which has rarely been mentioned in the literature of MATA, also plays a significant role in overall performance, especially in human-autonomy teaming under uncertain environments. On certain occasions during an operation, an agent might take an action that is considered sub-optimal or risky contrasting the previous experience and training. Modeling risk aversion in the agent decision-making process allows assessing the value of the behavior in different situations. Therefore, this chapter considers multiple sources of heterogeneity by varying agent capabilities and risk-averse factors in the agent decision-making process. The studies that explore the combined effect of multiple heterogeneity sources and provide parametric analysis of various heterogeneity factors on team performance are essential for designing the next generation of multi-agent systems in real-world applications.

Here, we develop an artificial intelligence framework for multi-agent heterogeneous teams to dynamically learn task distributions among agents and maximize the performance in complex operations through reinforcement learning. The heterogeneity of a multi-agent team in this content is described by 1) the difference in agent capabilities of task handling, sensing, and communication that have a direct impact on task-level transitions and perceived information accuracy and 2) the difference in agent decision-making process representing the level of risk aversion. The framework extends Dec-POMDP [94] to the heterogeneous teaming where agents are equally responsible for strategic planning and execution, and the heterogeneity in agent capabilities and decision processes are explicitly modeled. The proposed approach employs deep learning techniques [95] to improve computational efficiency and utilizes belief representation to summarize past experience allowing the incorporation of risk tolerance in the framework. Based on the proposed approach, we analyze how different levels and types of heterogeneity in a team influence the teaming performance during an operation. In addition, we introduce and incorporate some characteristics of humans in a team as a basis to facilitate studying human-autonomy systems within this framework.

The contributions of this work are:

- Development of a decentralized task allocation framework for teams with heterogeneous agents, which enables complex teaming interactions in environments with dynamic demands

and uncertainties

- Incorporation of risk aversion and perception accuracy in agent decision-making processes, and analysis of their effects on team performance in the presence of unforeseen events
- Quantitative investigation of the combined effects of heterogeneity and risk aversion on teaming performance

2.2 Methods

To formulate the multi-agent heterogeneous teaming problem, we assume agents are equipped with their own decision-making processes and are equally responsible for strategic planning and execution. No centralized coordinator commanding team members is considered. In addition, to ensure heterogeneity, agents have the same type of attributes but different capability levels and decision models. The proposed method inherits decentralized reinforcement learning as demonstrated in Fig. 2.1. With functional capabilities to handle tasks, sensing capabilities to perceive information, and communication capabilities to ensure information accuracy, all agents collaborate to achieve a common goal described by the reward system.

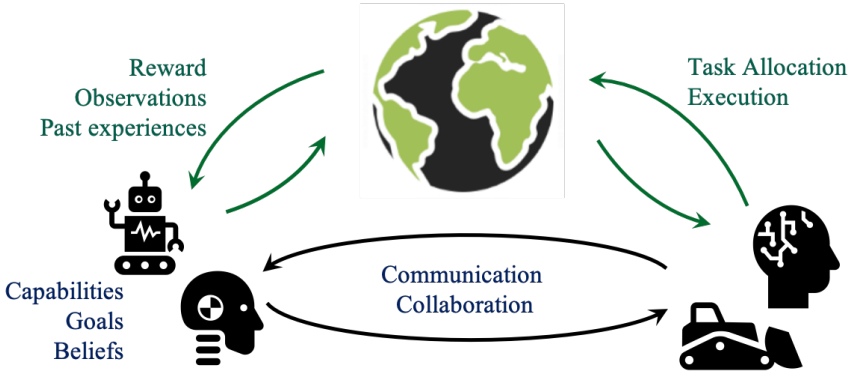


Figure 2.1: Schematic of decentralized reinforcement learning approach in heterogeneous multi-agent task allocation problems

2.2.1 Problem Formulation

The decision-making process for heterogeneous multi-agent teaming is formulated as an extended Dec-POMDP [94] where the capabilities of agents and information exchange are explicitly modeled. The proposed Heterogeneous Teaming Decentralized Partially Observable Markov Decision Process (HT Dec-POMDP) is defined as a tuple of $(G, \alpha, S, A, \mathcal{T}, R, Z, \mathcal{O}, M, \mathcal{I}, h, b^0, \gamma)$ where:

- $G := \{g_1, g_2, \ldots, g_p\}$ is a finite set of p tasks
- $\alpha := \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ is a finite set of n agents
- $S := S^D \times S^C, s \in S$, is the overall state that is factored into p tasks and described by task demand/severity levels $S^D := S_1^D \times S_2^D \times \cdots \times S_p^D$ and joint agent capabilities $S^C := S_1^C \times S_2^C \times \cdots \times S_p^C$
- $A := \times_i G, a \in A$, is the set of joint assignment decision by assigning α_i to g_j
- $\mathcal{T} := Pr(s'|s, a)$ is the state transition probability
- $R := f_R(s, a, s')$ is the reward function
- $Z := \times_i Z_i, z \in Z$, is the set of joint observations, where $z =< z_1, z_2, \ldots, z_n >$
- $\mathcal{O} := Pr(z|s', a)$ is the observation probability
- $M := \times_i M_i, m \in M$, is the set of joint information, where M_i represents the state information collected by agent α_i
- $\mathcal{I} = Pr(m|s', a, z)$ is the information probability
- h is the planning time horizon
- b^0 is the initial belief defined as the probabilistic distribution over the problem state space
- γ is the discount factor

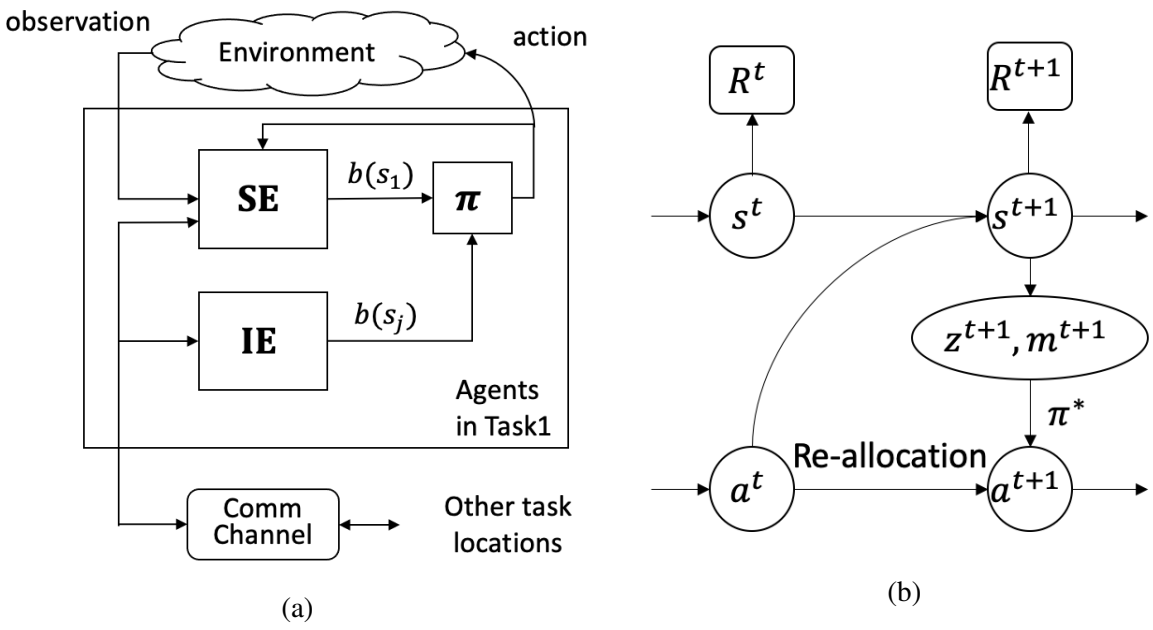


Figure 2.2: (a) Belief update mechanism for agents. For an agent assigned to task 1, the state estimator (SE) updates the belief $b(s_1)$, information estimator (IE) processes communicated observations and updates the belief over other tasks $b(s_j)$. The agent follows the policy π based on the overall belief updated for the next action. Comm Channel represents a communication strategy that can range from all-to-all to no communication. (b) Demonstration of state transition by time steps in team view. Allocation decisions for the next time step a^{t+1} are made based on agents' observations z and communicated information m received after the previous action a^t

HT Dec-POMDP provides a framework for heterogeneous teams to make decentralized decisions (Fig. 2.2a). The formulation splits the environment dynamics in the task domain, where the dynamics of each task can be individually modeled. However, the overall stochastic environment dynamics are maintained by considering conditional transition probabilities between tasks and task-specific capabilities, sensing-dependent observation probabilities, and communication-dependent information accuracy. Agents with higher sensing capabilities receive accurate information with higher probabilities, and the observations communicated by the agents with higher communication capabilities would be received accurately with higher probabilities. The state of the operation includes the severity level of each task S^D that evolves throughout the operation and the joint capability of the agents assigned to that task S^C depending on the pre-defined agent capabilities, i.e. $C_i = [c_1, ..., c_k]$ is the capability vector for agent α_i and k represents the number

of attributes. The environment reacts to agent actions depending on the joint capability of agents assigned to each task. As shown in Fig. 2.2b, during an operation, agents take actions a based on their belief over the system state and bring the state to another level in the next time step. Agents then receive partial observations z of task states through sensors, broadcast information m through limited communication, and receive reward R for accomplishing the tasks. Agents further update their belief with collected information using state estimation (SE) and information estimation (IE) described in Fig. 2.2a, and make decisions for the next action under policy π using the collected information.

The primary objective of this formulation is to find the optimal task allocation policy Π. Task allocation policy is defined as $\Pi : M \rightarrow A$, which maps the collected information M to decisions made by agents for the next action A. The optimal policy is defined as $\pi_* = argmax\{V(\pi)\}$ that maximizes team performance, where $V(\pi)$ represents the expected total future team reward defined as:

$$V_\pi = \mathbb{E}[\sum_{t=0}^{t=h} \gamma^t f_R(s^t, a^t)|\pi, b^0]. \tag{2.1}$$

Given the initial belief states b^0, the team follows the task allocation policy π for a time horizon of h steps. The reward is discounted by time (γ^t) reflecting that a higher reward is given if the team can accomplish all tasks earlier in time. We propose a Q-learning approach to learn the policy based on the agents' beliefs as discussed in Section 2.2.2.

2.2.2 Decentralized Deep Q-Learning with Beliefs

Dec-POMDP provides a decentralized decision model formulation but is known to be NEXP-hard. The optimal policy could be found by mapping each agent action-observation history to action, and the exact solution is only solvable in a finite time horizon [94]. In the context of multi-agent task allocation and the proposed HT Dec-POMDP framework, requiring agents to maintain information regarding all tasks through observations and communication makes the action-observation-information history very large, or even intractable, because a large time horizon is required usually for the team to complete the mission. The proposed approach extends the discrete

state to a continuous belief state space with a self-providing reward mechanism and allows agents to approximate the optimal solution in an infinite time horizon. The mechanism of self-providing rewards allows agents to generate rewards based on their own belief states instead of relying on the reward provided by the environment, which facilitates the modeling of heterogeneity in reward functions.

Belief state representation is helpful when solving a partially-observable environment [10, 94, 100]. A belief state is the probability distribution over the states of the environment identified by an agent. Instead of iterating all possible agent action-observation-information histories and extending the solution to an infinite time horizon, the agent's past experience is represented and summarized by a belief update mechanism. When the number of tasks and the number of task severity levels are large in complex operations, deep learning techniques could be used to learn the decisions based on beliefs directly.

$$
\begin{aligned}
b'(s') &= Pr(s'|m,z,a,b) \\
&= \frac{Pr(m|s',a,z)Pr(s'|a,z,b)}{Pr(m|a,z,b)} \\
&= \frac{Pr(m|s',a,z)Pr(z|s',a)Pr(s'|a,b)}{Pr(m|a,z,b)Pr(z|a,b)} \\
&= \eta Pr(m|s',a,z)Pr(z|s',a)\sum_{s} Pr(s'|a,s)Pr(s|a) \\
&\propto \mathcal{I}(s',a,z,m)\mathcal{O}(s',a,z)\sum_{s}\mathcal{T}(s,a,s')b(s)
\end{aligned} \tag{2.2}
$$

Let $b(s)$ denote the probability of an agent α_i being in the state s. The belief update follows Bayes' rule and can be computed by the state transition, observation, and information probabilities $\mathcal{T}, \mathcal{O}, \mathcal{I}$ defined in Section 2.2.1. The belief state inherits the Markov property since the updated belief does not depend on the previous belief states but only on the current state. Based on agent action a, observation z, and received information m, the updated belief state $b'(s')$ could be computed with Eq. 2.2, where $\eta = \frac{1}{Pr(m|a,z,b)Pr(z|a,b)}$ is considered as a normalizing factor which assures $\sum_{s'} b'(s') = 1$.

Being able to receive observations, communicate information, and maintain extracted state information locally in the belief space, the agents could learn the optimal task distribution in a completely decentralized fashion with the proposed HT Dec-POMDP formulation. The proposed deep Q-Learning approach shown in Algorithm 1 finds the optimal task allocation decision given the internal belief instead of being provided with the full observations of the environment [95]. The training algorithm is deployed to each agent, and agents learn the collaborative decision by interacting with the environment and storing past experiences in the form of beliefs.

Algorithm 1 Decentralized Deep Q-Learning with Beliefs

1: Initialize replay memories D and Q-Networks with random weights *for all agents*
2: **for** each training episode **do**
3: Initialize operation and beliefs *for all agents*
4: **for** each operation step **do**
5: ***For all agents***
6: Make decisions with ϵ-greedy and execute decisions
7: Receive observations from the assigned tasks and communicate observations in **Comm Channel**
8: Update beliefs with **SE** and **IE**
9: Compute reward and store transition into D
10: Train Q-Network with a randomly sampled minibatch of transitions in D
11: **end for**
12: **end for**

With processed observations and information in the state estimator and the information estimator (Fig. 2.2a), each agent updates the belief b as described in Eq. 2.3 (Alg. 1, Line 8), and the belief b summarizing the past observations and communicated information serves as the agent's state representation of the operation environment. Given a reward function directly on beliefs $f_R(b, a)$, the reward discount factor γ, and the time horizon h, the optimal action-value function $Q^*(b, a)$ is defined as the maximum possible expected total reward or Q-Value when having internal belief b and taking action $a = \pi(b)$ under the policy π as follows

$$Q^*(b, a) = \max_\pi \mathbb{E}[\sum_{t=t'}^{t=h} \gamma^{t-t'} f_R(b^t, a^t) | b^t = b, a^t = \pi(b)]. \tag{2.3}$$

Then, the optimal policy can be retrieved by $\pi^*(b) = \text{argmax}_a Q^*(b, a)$. The optimal Q-Value

$Q^*(b, a)$ has to satisfy the *Bellman Optimality* to maximize the expected Q-Value of the next belief-action pair (b', a'):

$$Q^*(b, a) = \mathbb{E}_{b'}[f_R(b, a) + \gamma \text{max} Q^*(b', a')|b, a]. \tag{2.4}$$

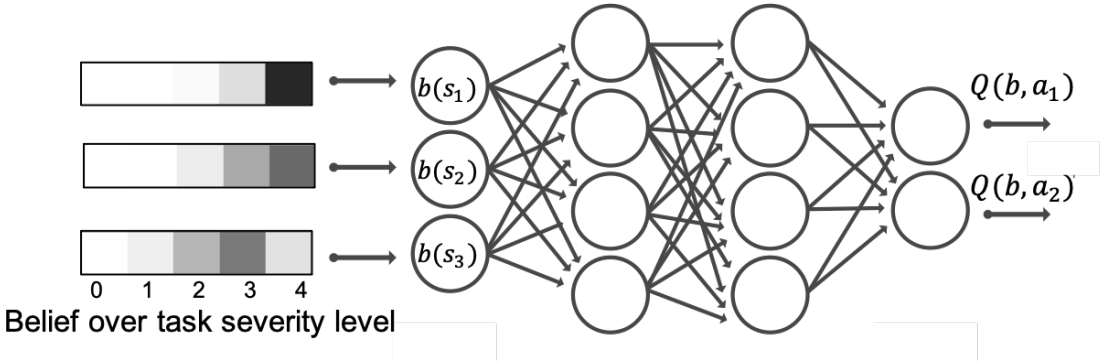
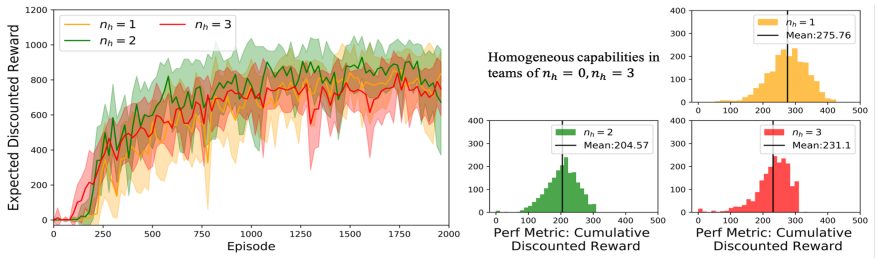


Figure 2.3: Schematic of a Q-Network with an agent's belief over severity levels of 3 tasks as input and Q-Values of 2 actions as output. In the proposed approach, the agent's belief over task severity levels are used as an input to the Q-Network and the outputs are the Q-Values in the action space.

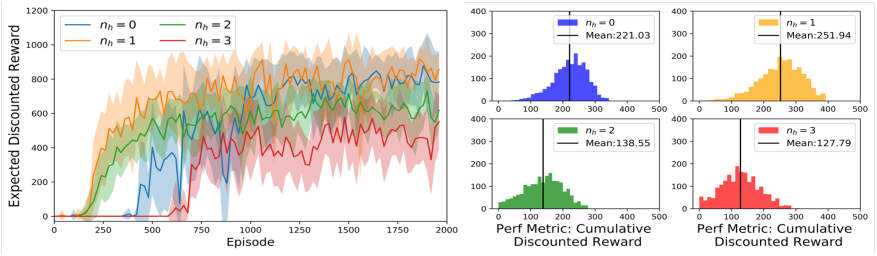
As shown in Fig. 2.3, we use a neural network to approximate the Q-Value $Q(b, a; \omega)$ with belief states as network inputs and train the Q-Network for each agent so that $Q(b, a; \omega) \approx Q^*(b, a)$. The Q-Network takes the belief state as input and outputs the Q-Value corresponding to each action. Given transitions $< b, a, b', f_R(b, a) >$ (Alg. 1, Line 10), the Q-Network is trained by minimizing the loss function $L_i(\omega_i)$ between predicted Q-Value $Q(b, a; \omega_i)$ and the target Q-Value g_i for each iteration i, where $g_i = f_R(b, a) + \gamma \text{max} Q(b', a'; \hat{\omega}_i)$ and $\hat{\omega}_i$ is the target network parameter which is updated less frequently:

$$L_i(\omega_i) = \mathbb{E}_{b,a,f_R,b'}[g_i(b', a', \hat{\omega}_i) - Q(b, a; \omega_i)]. \tag{2.5}$$

The choice of reward function is an important factor in the reinforcement learning process, and finding the optimal reward function can expedite the training process [101, 102]. In our task allocation framework, agents receive an ultimate high reward upon mission completion (all task severity levels are zero), while a comparably small reward is given to agents for partially completing the mission (only a fraction of tasks are completed). This reward structure can incentivize agents to



(a)



(b)

Figure 2.4: Learning curves with shaded area indicating standard deviations of every 20 training episodes and performance evaluations with 2000 runs for different team compositions with (a) heterogeneity in capabilities only (b) heterogeneity in capabilities and decision models

reach mission completion during training. In particular, we define the reward function based on the agent's belief as:

$$f_R(b) = W_l^T R_l \tag{2.6}$$

The level completion reward R_l is the mission goal reward describing how much reward is received at each task severity level l. The level completion weight is defined as a function of belief with two tuning parameters: $W_l = f(b; w, b_{th})$, where $w \in [0, 1]$ is the weight for partial completion, representing how much agents reward themselves with partial credits and $b_{th} \in [0, 1]$ is belief reward threshold, deciding how generous agents reward themselves since some agents want to be more sure about the situation in order to reward themselves. Here, agents receive up to 30% of the maximum reward when up to 75% of the tasks are completed. They receive 100% of the reward upon the completion of all the tasks. In addition, the effect of variation in the belief reward threshold as a risk factor in the decision process is investigated in detail in Section 2.3.2.2.

2.3 Results

We showcase our analysis on a disaster relief scenario which is an extension of the Firefighting domain, a benchmark used in the evaluation of Dec-POMDP planning algorithms [94]. In this new scenario, a team of n agents has to extinguish fires in p houses and safely rescue people trapped in the burning houses. The fire extinguishing and rescue demands can vary in the range of $[0, 4]$, where 4 is the worst-case scenario and zero represents that the task is completed. Each agent has its unique capabilities in extinguishing fire and rescuing people. In particular, we define four attributes for each agent, namely fire extinguishing, rescue, sensing, and communication capabilities. We consider the capability level for each attribute as a discrete value in the range of $[0, 5]$, where higher values represent better agent capability and zero represents that the agent is incapable of an attribute. The higher the joint capability of the agents assigned to the same task is, the higher the probability of the demand for that task being reduced, which implies agents have to collaborate to resolve the worst situation with the highest probability. Higher sensing and communication capabilities help the agents to approximate the environment status with higher accuracy. Agents would learn to cooperate on the same task when task demand levels are high and synergistically work on different tasks at the same time when task demand levels are low to resolve the severe situation faster. They might also decide to stay at certain locations to prevent the situation from getting worse.

2.3.1 Effect of Heterogeneity on Team Performance

2.3.1.1 Heterogeneity in Capabilities

In a heterogeneous team, agents may be specialized and have different capability levels. To study the effects of heterogeneity in agent capabilities on team performance, we consider different team compositions. To ensure fairness on team capabilities, the mean capability levels for firefighting and rescue attributes are kept at 2. For the same attribute, when there is an agent with lower capability, there is another agent in the same team with higher capability to keep the mean con-

stant. In order to focus on the heterogeneity of task-specific capabilities, the sensing capability and communication capability of all agents in all teams are fixed at level 2. With a team of three agents and four attributes (i.e. firefighting, rescue, sensing, communication) mentioned above, there are four possible team compositions represented by $n_h = 0, 1, 2, 3$, where n_h represents the number of agents with lower than average capabilities. For instance, in the case of $n_h = 1$, one agent has firefighting capability 1 and rescue capability 1, and the other two agents have capabilities ≥ 2 to keep the mean capability of each attribute as 2. Note that both $n_h = 0$ and $n_h = 3$ teams have homogeneous capabilities where $C_i = [2, 2, 2, 2]$ for all agents in order to satisfy the above conditions.

Fig. 2.4a shows the learning curves and performance evaluations for team compositions with different capability distributions assuming all agents decide rationally. After training, the performance of each team is evaluated in 2000 operation trials shown in the histograms. The performance metric is the discounted reward/score given by the environment. This metric indicates how effectively a team completes all tasks in execution. Higher performance indicates the team is able to reduce all task severity levels to zero within fewer time steps. Note that the reward during training is obtained using agent belief, which is different than the performance metric, but it still reflects the team's performance. Results show that heterogeneity in agents' capabilities affects the overall performance of the team. In the simulated experiments, for the heterogeneous team of $n_h = 1$ the team has the best performance based on the mean cumulative discounted reward achieved by the team in 2000 evaluations. When agents have specialized capabilities ($n_h = 2$) or uniform capabilities ($n_h = 0$ or 3), the team performance deteriorates.

2.3.1.2 Heterogeneity in Capabilities and Decision Models

Next, we study the effect of sub-optimal decision-makers on the performance of studied team compositions of the previous example. Agents and in particular humans in the team might not always make the optimal (rational) decision since sometimes they are risk-takers and tend to make sub-optimal decisions. This behavior is called noisy rational decision, and it could represent hu-

man creativity in learning and execution. In the previous example, we assume that the agents with relatively lower capabilities (represented by n_h) would make less rational decisions resulting in heterogeneity in decision-making across the team. We consider these agents as human-like agents assuming that humans have less task-handling capabilities but their creativity and higher risk tolerance lead to noisy rational decisions. To model noise in agent decisions, we incorporate a noisy rational model, a widely used human decision model in cognitive science [31–33], into our proposed framework. In particular, we give an agent the option to take any action with certain probability defined as

$$Pr(a|b) = \frac{exp(\theta \cdot Q(b,a))}{\sum_{a_i \in A_i} exp(\theta \cdot Q(b,a_i))}, \tag{2.7}$$

where A_i denotes the action space for an agent. The noisy rational model takes the output of Q-Network (Q-Value $Q(b,a)$ described in Section 2.2.2) and computes the probability of taking each possible action a given a belief state b. The tuning parameter $\theta \in [0, \inf]$ determines how rational the agent makes the decision. When $\theta = 0$, agents make completely random decisions; when $\theta \to \inf$, an agent makes rational decisions, i.e., chooses the action with the highest Q-Value based on its Q-Network. A noisy decision is sub-optimal with the trained network but it could be a better decision if training is insufficient.

With added noisy rational decisions to agents with lower than average capabilities, we observe that in the training curves in Fig. 2.4b, the teams with homogeneous capabilities (i.e., $n_h = 0$ and $n_h = 3$) start to get reward much later than the other teams, implying that homogeneous teams or a team where all its agents often make sub-optimal decisions require longer training. In the performance evaluation, the team with $n_h = 1$ still has the best performance among all team compositions highlighting that forming a specialized or a uniform team does not lead to the best training efficiency and performance.

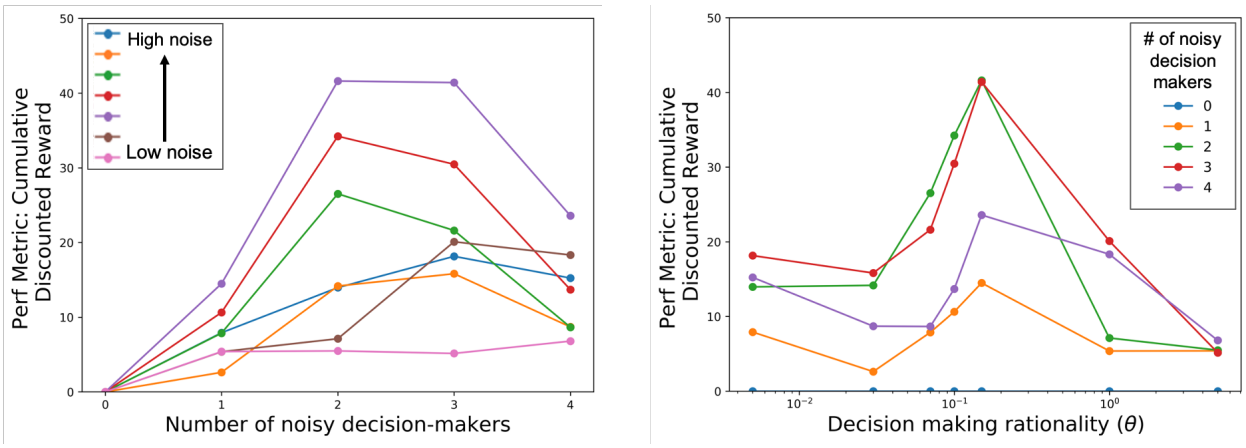
The results of the analysis in this Section show how a heterogeneous team can outperform a homogeneous one in complex operation scenarios. The proposed framework is capable of capturing the deficiencies in the training for different teams and the uncertainties in the team performance and provides insights about the optimal selection of a team of agents leading to stable training and

reliable performance.

2.3.2 Effect of Risk Aversion on Team Performance

2.3.2.1 Auxiliary Tasks

It is often challenging to deploy the trained agents in a real operation where they face events that they are not trained for. We define auxiliary tasks to simulate unexpected events during operation, i.e. tasks that are present in the operation scenario but were not present in training. In circumstances where agents encounter auxiliary tasks, the proportion of sub-optimal decision-makers in the team as well as the level of noise in the agent's decisions can affect the team performance significantly. Here, we evaluate the effect of facing risks in agents' decision-making process when tackling this unexpected situation. Such a scenario is analogous to a team including humans who have different levels of creativity and/or uncertainty in their decisions. In this study, we explored the effect of heterogeneity in decision-making among the team members on completing a mission including tasks that were not present during the training (i.e., auxiliary tasks). We considered 6 regular tasks, and 4 agents, and added 2 auxiliary tasks in training to explore the ability of the team to adapt to unforeseen demands in performance evaluation. Auxiliary tasks remain at zero level during training and start at the highest level in evaluation. Fig. 2.5a shows the performance of different team compositions when the number of noisy (sub-optimal) decision-makers in the team is varied. In addition, Fig. 2.5b shows the performance of each team by varying the level of noise in the agents' decision-making capabilities. Results show that there is a trade-off between the number of noisy decision-makers and the level of rationality. In particular, it is observed that there exists an optimal value for both the number of noisy decision-makers in the team and the level of noise in their decision-making processes which result in maximum reward and the best performance. A carefully designed team including risk-taker agents enables the team to handle unforeseen circumstances and make up for the deficiencies of the Q-Network by occasionally working on auxiliary tasks or tasks assumed to be sub-optimal choices under the Q-Network.



(a) (b)

Figure 2.5: (a) Effect of number of noisy decision-makers in a team on the team performance, (b) effect of decision-making rationality on the team performance. Each point represents the mean value of 1000 evaluations.

2.3.2.2 Risk Tolerance/Award Generosity

In a decentralized decision-making process, agents obtain rewards based on their evaluation of the environmental status after each action. Such a judgment is made based on agents' belief identifying how certain an agent thinks about that a task is completed. In a heterogeneous team, different agents (e.g. humans and autonomous agents) might have different levels of standard to label a task as completed. Here, we investigate the effect of the belief threshold, which is another risk factor in the decision-making process, on the team performance.

We considered a team of three agents with similar capabilities in the disaster relief scenario. The belief reward threshold of the agents is varied from 10% to 99% and the team performance is evaluated for each case. A $p\%$ belief threshold means that the agent considers a task to be complete if it is at least $p\%$ certain about that. An agent considers a mission completed when its belief in the completion of all the demands (i.e., fire and rescue levels at all locations) falls within its belief threshold. The performance evaluation includes 1000 realizations with random but heavy (≥ 3) initial demand levels. We investigate two performance metrics; the cumulative discounted reward and the first completion step of all tasks (Fig. 2.6).

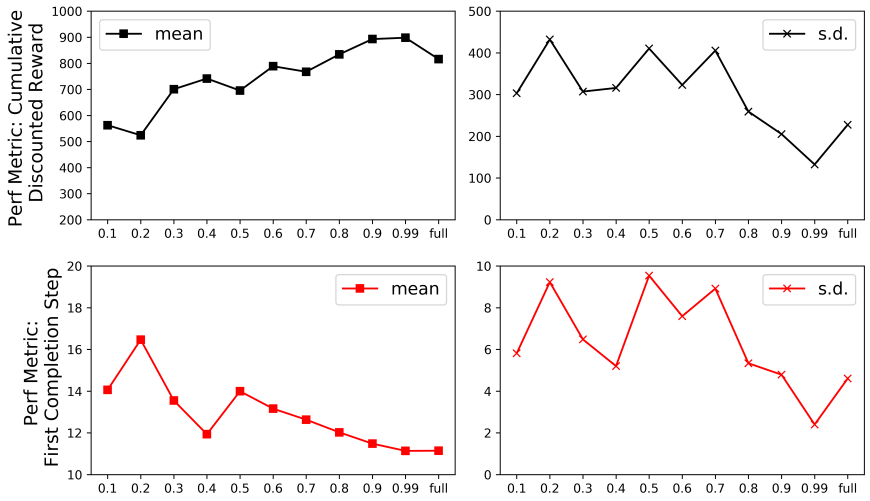


Figure 2.6: Effect of belief reward threshold (b_{th}) on team performance, i.e., cumulative reward (top row) and first completion step of all tasks (bottom row). Results are compared to cases when the system provides a reward on full completion (full) of the tasks. The mean value and their corresponding standard deviations are obtained from 1000 simulations for each case.

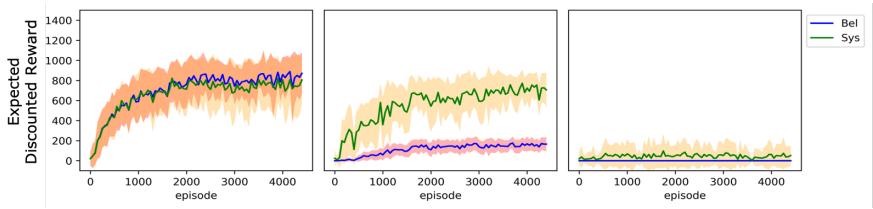


Figure 2.7: Low (left), medium (center), and high (right) levels of added uniform prior in training. Blue lines indicate the average agent belief reward and green lines indicate the corresponding reward system would have given. Solid lines and shaded areas represent the mean value and standard deviations of every 20 episodes during the training.

Results show that the belief reward threshold of $b_{th} = 0.9, 0.99$ leads to the best performance based on the received cumulative discounted reward. The performance outperforms the case of system reward, i.e., when the reward is given to the agents by the system during learning instead of their belief over environment status. In addition, there is a trade-off in the belief threshold. With a large threshold, agents receive rewards for accurate state inference and therefore significantly improve the performance uncertainty represented by standard deviations of the evaluated rewards. With small thresholds, however, agents receive rewards more frequently to help learning, but the performance uncertainty is significantly larger than high belief thresholds, meaning that it is risky

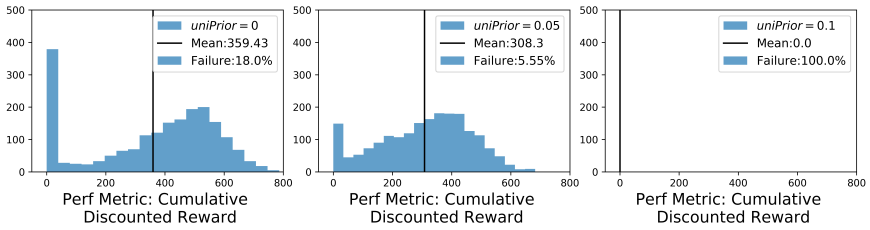


Figure 2.8: Low (left), medium (center), and high (right) levels of added uniform prior in performance evaluation with a sudden event.

to select an agent that is generously rewarding itself in the team, and the performance uncertainty is not guaranteed.

2.3.2.3 Balancing Past Experience and Recent Observations

In real operations, a team of agents usually encounters sudden events that have not been experienced during training. Making decisions purely based on past experience leads to reluctant behaviors in reaction to sudden events. The incapability of handling these situations might result in failure to complete the defined tasks for the agents. Inspired by human adaptivity and the ability to promptly assess the current environment status, the agent's ability to balance past experience and recent observations is modeled in the developed framework. In particular, we define a parameter u called uniform prior that adjusts belief probability distributions before receiving observation as a tuning parameter. Such a parameter identifies to what extent an agent takes the risk of trusting recent observations rather than past experience in its decisions for the next action.

Figure 2.7 shows the training curves with three levels of added uniform prior. When a uniform prior value $u \in [0, 1]$ is high and is added to the belief probability, by re-normalizing the probability, the agent belief gets proportionally biased toward u before taking observation and communication information. Higher u indicates an agent would rely more on the upcoming information to make decisions. Results show that there is a trade-off between the effect of current observations and past experience on decision learning and the performance of a team. A higher prior, when agents base their decisions more on recent observations, results in the worst training quality in regular operations. In Fig.2.8, trained agents are evaluated in an operation with a sud-

den event that is not present during the training. Particularly, we increase the fire levels of the houses to their maximum level after ten-time steps since the start of the simulations. The agents are expected to complete the tasks, i.e., bring all fire levels to zero and rescue all people, in 30 time steps; otherwise, the operation is considered as failed. We observed that a medium level of uniform prior could help the team react to such sudden events. The failure rate of completing all tasks is noticeably lower than the case when agents base their decisions purely on past experience or recent observations.

2.4 Conclusions

In this chapter, we developed an artificial intelligence framework for multi-agent heterogeneous teams to dynamically learn task distribution and maximize performance in complex operations through reinforcement learning. The proposed framework HT Dec-POMDP is compatible to model various sources of heterogeneity within a team, captures aspects of intelligence that produce collaborative teaming, and provides the opportunity to quantitatively investigate the effects of heterogeneity and risk aversion on task allocation in multi-agent systems. Results of this study show that a well-designed heterogeneous team significantly improves team performance, and including agents with higher risk tolerance in a team significantly improves the chance of achievement in uncertain environments.

The proposed framework can be used as a basis for further developments and design of multi-agent systems, particularly human-autonomy teams as one of the future works, where forming a team composed of agents with heterogeneous capabilities and different levels of creativity and risk tolerance is inevitable. Note that improving the computational efficiency of existing algorithms might be required particularly when more types of tasks and larger teams are involved. Nevertheless, the proposed approach facilitates accounting for information disruption and risky behaviors and can provide insights into making high-level decisions in complicated operations. Moreover, with the help of deep learning approaches and defined reward functions, additional types of information can be included in the belief, as the operational scenario may require. Examples include

information regarding previous assignments and consideration of the cost of traversing between task locations for spatially distributed demands.

CHAPTER 3

Task Allocation with Load Management in Multi-Agent Teams

This chapter builds upon Chapter 2 to enhance the behavior learning of autonomous teammates to achieve multiple objectives including completion efficiency and load reduction.

3.1 Introduction

Multi-agent systems have significant potential in generating management-level strategies for real-world large-scale operations such as humanitarian relief, search and rescue, transportation, warehouse management, and various other scenarios that involve the teaming of heterogeneous agents or human-autonomy collaboration. In a team of collaborative agents, instead of being managed by a centralized system, decentralized agents have equal authority in task planning and execution, ensuring operation effectiveness, team resilience in unexpected situations, and scalability. With the help of emerging computing systems, recent advances in multi-agent task allocation [9, 103, 104] have deliberated on distributed systems and deep reinforcement learning methods [22, 23, 27, 94, 95] for complex scenarios in addition to centralized approaches [10, 15, 28]. Most coordination and collaboration strategies for efficient task allocation [105] define the objective as accomplishing tasks [27] or reaching task locations [22, 28] in fewer steps. Although these approaches could sufficiently accomplish the problem objective, there is a lack of consideration of risk management since unanticipated situations might require additional strategic amendments in real-world operations. Agents in a team should be intelligent enough to prepare for the unknown.

To mitigate such risks and potentially allow agents to be available for unexpected events, some

agents should idle. Being constantly active would overload the agents, would be especially harmful to human agents, and would cost extra resources. In the literature, load is handled as management or load allocation problems [106–108] in the field of multi-agent computing. In human-robot collaboration, the load usually refers to the workload measured by human heart-rate variability [25], serving as feedback to reduce stress and improve overall team performance [26]. In the context of multi-agent task allocation in dynamic environments, the idling time is a performance measure to indicate task load balance [29], a precondition/constraint to perform other tasks [21], or a time indicator for inoccupation [30]. Limited research has explored the effect of idling as an option and has investigated the effects of load reduction on performance for multi-agent teams. Therefore, this work considers idling behaviors as an incentive to reduce task load and save resources represented by agent capabilities and task reassignment.

In this chapter, we leverage our previous work [27] on task allocation for heterogeneous multi-agent teams to model task load management based on agent preference in decision-making as a secondary objective along with maximizing team performance. Multi-agent teams with reduced load respond more reliably in emergencies that require extra load/resources or in situations where one of the team members is compromised. The proposed work also accounts for individual agent preferences in task assignments. The analysis of the impact of load reduction and the importance of collaboration highlights the benefits of the proposed decision-making framework.

The contributions of this work are:

- Efficient learning of load reduction in multi-agent teams with heterogeneous capabilities without compromising team performance
- Embedding agent preferences in decision-making by customizing individual reward functions
- Inferring team resilience by characterizing the importance of each agent on team performance.

3.2 Methods

To enable intelligent teaming of heterogeneous agents in task allocation problems, it is assumed that all agents possess individual decision-making processes instead of following commands. In lieu of being subordinate members, agents are equally responsible for strategic planning and task execution. We characterize the heterogeneity in agent decision models by allocation preferences which can be customized by agent reward functions. While agents have the same type of attributes to handle tasks, sense the environment, and communicate information, the capability levels for each attribute are heterogeneous. Such heterogeneous and specialized agents need to collaborate to complete all the tasks within fewer steps. An efficient learning method is desired to achieve intelligent teaming behaviors with load management, by which agents stay idle when they are not needed without compromising the overall teaming effectiveness.

3.2.1 Problem Formulation

The decision-making process for a team of heterogeneous agents with load management inherits the formulation of our previous work on heterogeneous teaming decentralized partially observable Markov decision process (HT Dec-POMDP) [27] and incorporates two additional features allowing intelligent load management: 1) agent's option to idle, and 2) penalty on unnecessary task assignment depending on capabilities and current task assignment. The modifications of the proposed framework heterogeneous teaming with load management (HTLM Dec-POMDP) based on HT Dec-POMDP include:

- $\bar{G} := \{g_1, \ldots, g_p, \bar{g}_1, \ldots, \bar{g}_q\}$ is a finite set of p tasks and q idle locations
- $\alpha := \{\alpha_1, \ldots, \alpha_n\}$ is a finite set of n agents
- $\bar{A} := \times_i \bar{G}, a \in \bar{A}$, is the set of joint assignment decision by assigning α_i to g_j
- $\bar{R} := f_R(a_o, b(s), a)$ is the reward function, where a_o records the previous assignment, b is the belief defined as the probabilistic distribution over the problem state space.

This formulation follows the fundamental modeling of environment dynamics in HT Dec-POMDP [27], and explicitly defines various types of attributes (task-specific, sensing, and communication capabilities) for agents and the corresponding probabilities that describe the stochastic environment and the agent's perception of the environment and information. In many studies, agents stay active throughout the operations, which might lead to excessive usage and risk exposure. Instead, HTLM Dec-POMDP provides agents with the option to stay idle by including idle locations as part of the assignment decisions described by $\bar{G}$ and $\bar{A}$. In addition, the reward function $\bar{R}$ not only defines the rewards for completing the tasks, but it also penalizes unnecessary assignments and reassignments to induce agent idle behaviors without compromising team performance. Since the formulation is decentralized, agents have their own reward functions representing their understanding of the situation and preferences on the task assignment, which might be heterogeneous among the team members.

The goal of the formulation is, for each agent, to find an optimal series of task allocation actions or policy π_* that maximizes the operation performance so that $\pi_* = argmax\{V(\pi)\}$. The performance is represented as the policy value $V(\pi)$ defined as the expected total future discounted reward.

$$V_\pi = \mathbb{E}[\sum_{t=0}^{t=h} \gamma^t f_R(a^{t-1}, b^t(s), a^t)|\pi, b^0]. \tag{3.1}$$

To give more incentives for agents to accomplish tasks at earlier time steps, the reward is discounted by time (γ^t). At the beginning of an operation, each agent has an initial belief $b^0(s)$ over the operation state. At each time step of a time horizon h, agents execute decisions $\bar{A}$, receive partial observations Z, and receive information from others M. After experiencing the action-observation-information histories, agents update their beliefs over the state following Eq. (3.2) with capability-dependent transition probability $\mathcal{T}$, observation probability $\mathcal{O}$, and information

accuracy $\mathcal{I}$ [27] in order to make the decisions for the next time step, namely

$$\begin{aligned} b'(s') &= Pr(s'|m, z, a, b) \\ &\propto \mathcal{I}(s', a, z, m)\mathcal{O}(s', a, z) \sum_{s} \mathcal{T}(s, a, s')b(s). \end{aligned} \tag{3.2}$$

The approaches of defining the reward functions and learning the optimal policies based on beliefs are discussed in Sections 3.2.2 and 3.2.3

3.2.2 Embedding Agent Preference in Reward Shaping

Defining a reward function, also referred to as reward shaping, plays a critical role in reinforcement learning. A carefully designed reward function can expedite the training process while satisfying the problem objectives. In our decentralized task allocation problem, there are two objectives: 1) mission completion and 2) load management. Moreover, reward functions could be customized to model agent preferences, meaning that agents can have different reward functions depending on their willingness to take risks and to reassign tasks. Providing rewards based on the actual operation state [21] or system score [95] is not applicable since there does not exist a centralized informer. Therefore, agents reward themselves based on their belief in the status of the demands and the environment.

$$f_R(a_0, b(s), a) = R_{comp} + R_{LM}, \tag{3.3}$$

$$R_{comp} = W_l^T R_l, \tag{3.4}$$

$$R_{LM} = R_{II}(b(s)) + T_{RP}(a_0, a). \tag{3.5}$$

Eq. (3.3) defines the reward function for the proposed HTLM Dec-POMDP. The first part shown in Eq. (3.4) is the same as in HT Dec-POMDP. Deciding how much and how generous each agent informs the reward of mission completion based on partial completion weight ω and belief reward threshold b_{th} respectively, where $W_l = f(b(s); w, b_{th})$ and R_l provides the reward for

each task severity level l [27]. With this reward function, the team of agents learns to accomplish the desired tasks effectively, and there exists a number of optimal solutions. The deep learning approach randomly converges to one of the optimal solutions but fails to avoid task overload. The second part of the reward function is the reward of load management R_{LM} in HTLM Dec-POMDP defined in Eq. (3.5). R_{LM} has two features - idle incentive R_{II} and reassignment penalty T_{RP} - in order to achieve the following behaviors without compromising operation effectiveness: idling as much as possible while avoiding excessive capability usage and unnecessary task reassignment. Load management serves as the secondary objective after mission completion. Therefore, the maximum possible R_{LM} is much smaller than R_{comp}. Additionally, R_{II} is positive (reward) if the agent is assigned to one of the idle locations and negative (penalty) if the capability level of the agent is much higher than the believed severity level for the assigned task, namely being too capable of handling the task. The reassignment penalty T_{RP} is a cost matrix defining the cost of changing task assignment from a_0 to a, and sometimes this could be the traversing cost matrix between each location if there is no cost of switching tasks within the same location.

In the task set $\bar{G}$, there are idle locations corresponding to the task locations. We assume that there is no cost of going into and out of idling within the same location, and there is a penalty for being reassigned to a different location. A sample R_{RP} cost matrix for 2 locations is shown in Table 3.1, where the traversing cost between two locations is $tc \leq 0$.

Table 3.1: Two-Location Traversing Cost Matrix

	Loc 1 Task	Loc 2 Task	Loc 1 Idle	Loc 2 Idle
Loc 1 Task	0	tc	0	$-\infty$
Loc 2 Task	tc	0	$-\infty$	0
Loc 1 Idle	0	tc	0	$-\infty$
Loc 2 Idle	tc	0	$-\infty$	0

If only one idling action is considered for multiple locations with no cost of going into idling and out-of idling, agents can bypass the reassignment penalty by going idle and being assigned to a different task with two steps, which has an indirect impact on team effectiveness. Including

multiple idle locations could explicitly define the reassignment penalty and have a direct impact on the agent's decision for each time step. To avoid the additional computation due to the larger action space, we generate a reduced action space for each agent $\hat{A} = \{g_1 \dots g_p, \bar{g}_o | a_o\}$ to restrict idling to a different location. Only the idle location $\bar{g}_0$ that has the same location as the previous assignment a_0 is included in the reduced action space while other idle locations are eliminated. For example, as shown in Table 3.1, the actions with ∞ cost would be eliminated given the previous assignment. The reduced action space for each agent at each step is $p + 1$, where p is the number of tasks and the additional action is idling at the same location. Note that by removing idle locations and setting $R_{LM} = 0$, the framework would reduce to HT Dec-POMDP. It is intriguing to investigate the effect of R_{LM} on team behaviors and performance when idle actions are available, which is discussed in Section 3.3.1.

3.2.3 Decentralized Deep Q-Learning for Load Management

To achieve load management by avoiding excessive capability usage and unnecessary assignment, an agent has to acknowledge the severity of the operation and record the previous assignment. To approach the optimal solution, the proposed method extends the decentralized deep Q-learning with beliefs [27] as shown in Algorithm 2 (underlined parts indicate the contribution of the proposed learning method for load management), and an agent makes intelligent decisions based on a knowledge state $\mathcal{K} = \{b(s), a_o\}$ which consists of its belief of the operation state $b(s)$ and the previous assignment a_o. The knowledge state update (Alg. 2 Line 8) inherits the Markov property since the next belief $b'(s')$ only depends on the current belief $b(s)$ as described in Eq. (3.2), and the decision made becomes the previous assignment in the next time step.

Given the reward function discussed in Section 3.2.2, the discount factor γ, and the time horizon h, the optimal action-value or Q-Value $Q^*(\mathcal{K}, a)$ is defined as the maximum possible expected total reward when having knowledge state $\mathcal{K}^t = \mathcal{K} = \{b, a_o\}$ and taking action $a = \pi(\mathcal{K})$ under policy π.

$$Q^*(\mathcal{K}, a) = \max_{\pi} \mathbb{E}[\sum_{t=t'}^{t=h} \gamma^{t-t'} f_R(\mathcal{K}^t, a^t) | \mathcal{K}^t = \mathcal{K}, a^t = \pi(\mathcal{K})]. \tag{3.6}$$

Algorithm 2 Dec Deep Q-Learning for Load Management

Initialize replay memories D and Q-Networks with random weights *for all agents*
for each training episode **do**
  Reset operation and knowledge state $\mathcal{K}$ *for all agents*
  for each operation step **do**
    For all agents
    Select decisions from $\hat{A}$ with ϵ-greedy and execute
    Observe and communicate observations
    Update $\mathcal{K} = \{b(s), a_o\}$
    Compute $f_R(\mathcal{K}, a)$ and store $\mathcal{X}$ into D
    Train Q-Network with a randomly sampled minibatch of transitions in D
  end for
end for

Under *Bellman Optimality*, the optimal Q-Value $Q^*(\mathcal{K}, a)$ can then be rewritten using the next knowledge-action pair $(\mathcal{K}', a')$.

$$Q^*(\mathcal{K}, a) = \mathbb{E}_{\mathcal{K}'}[f_R(\mathcal{K}, a) + \gamma \text{max} Q^*(\mathcal{K}', a') | \mathcal{K}, a]. \tag{3.7}$$

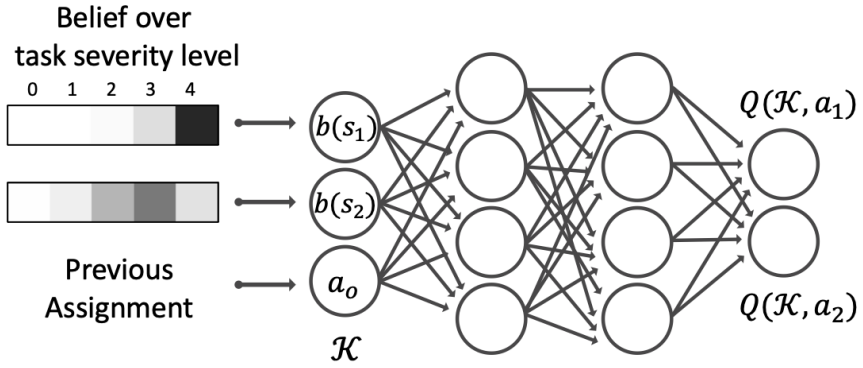


Figure 3.1: Schematic of a Q-Network with an agent's belief over severity levels of 2 tasks and previous assignment as input and Q-Values of 2 actions as output. In the proposed approach, the agent's knowledge state $\mathcal{K}$ is used as an input to the Q-Network, and the outputs are the corresponding Q-Values $Q(\mathcal{K}, \cdot)$ in the action space.

Instead of exploring the full continuous knowledge state space, a neural network as shown in Fig. 3.1. The Q-Network $Q(\mathcal{K}, a; \theta)$ is used to approximate the optimal Q-Value $Q(\mathcal{K}, a; \theta) \approx Q^*(\mathcal{K}, a)$. The optimal policy for each agent can then be computed by $\pi^*(\mathcal{K}) = \text{argmax}_{\hat{a}} Q^*(\mathcal{K}, a)$. An agent selects the action with the highest Q-Value from the reduced action space $\hat{A}$. Recall that

$\hat{A} \subseteq \bar{A}$. The input of the Q-Network is the knowledge state which includes the belief state over all task severity levels in probability space and the one-hot encoded previous assignment. The total number of input neurons is $pL + |\bar{G}|$, with p being the number of tasks and L being the number of task severity levels. The output of the Q-Network is the Q-Value for each action, and the size of the output is $|\bar{G}|$. The Q-Network output includes all tasks and idle locations since agents can be assigned to any of them throughout an operation. The reduced action space is helpful when generating random actions and selecting the optimal action (Alg. 2, Line 6) so that agents do not waste effort on reassigning to the location with ∞ cost. The benefit of using a deep learning approach in decentralized multi-agent teams is that the state space and the actions space are linear with respect to the number of tasks.

During each training iteration i, a set of transition data $\mathcal{X} =< \mathcal{K}, a, \mathcal{K}', f_R(\mathcal{K}, a) >$ is collected into replay memories (Alg. 2, Line 9). With the Bellman Eq. (3.7), the Q-Network $Q(\mathcal{K}, a; \theta)$ should converge to a target Q-Value $y_i = f_R(\mathcal{K}, a) + \gamma \text{max} Q(\mathcal{K}', a'; \hat{\theta}_i)$, where the parameter of the target Q-Network $\hat{\theta}_i$ is updated with θ less frequently. The difference in Q-Value and the target Q-Value is the loss $\mathcal{L}_i$ to be minimized to reach Bellman Optimality as defined in:

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{\mathcal{X}}[y_i(\mathcal{K}', a', \hat{\theta}_i) - Q(\mathcal{K}, a; \theta_i)]^2. \tag{3.8}$$

3.3 Results

We analyze the teaming behaviors of the proposed HTLM Dec-POMDP framework in a disaster relief operation scenario. The operation has a team of 5 agents and 2 disaster sites. At each site, there are two different but conditionally dependent tasks: extinguishing fire and rescuing people from the burning site (4 tasks in total). The demand/severity level for each task is within the range of $[0, 4]$. At the beginning of the operation, all tasks start with a severity level of 4, and the operation is considered complete when the severity levels of all tasks are 0. Each agent is equipped with two task-specific attributes (firefighting and rescue) and two perception-related attributes (sensing and communication). The joint capability level for each attribute is within the range of $[0, 5]$, and

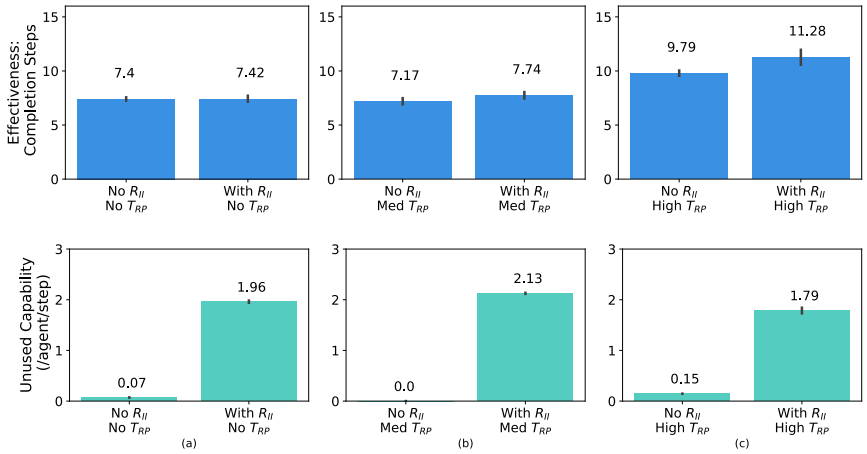


Figure 3.2: Effect of idle incentive R_{II} on operation effectiveness and capability usage with (a) no reassignment (b) medium reassignment and (c) high reassignment penalty T_{RP}. The performance evaluation includes 500 realizations, and each bar plot shows the mean and the standard error.

it is assumed that joint capability is computed as the addition of the capabilities of the agents assigned to the same task and capped at level 5. Higher task-specific capabilities would reduce the task demand with higher probabilities, and task levels would not increase once they reach level 0. The dependency between tasks is also modeled in task transition probabilities, where the probability for rescuing demand to reduce is 0 when fire level is high (> 2) no matter how capable the agents are. An agent is only able to observe the task level at the assigned task, making the problem partially observable. Higher sensing and capabilities would help agents perceive the environment more accurately, and the observed information broadcasted by the agents with higher communication capabilities would be trusted more. Here, the sensing and communication capabilities for all agents are set to level 3 (moderately high accuracy) since perception accuracy is not the focus of this study. With a properly designed reward function, agents would reduce the task demands efficiently by cooperating on the same task or simultaneously working on different tasks. Meanwhile, without compromising the operation effectiveness, agents' load is reduced by appropriate idling behaviors and avoiding extra capability usage and unnecessary assignments.

3.3.1 Effect of Idle Incentive and Reassignment Penalty on Load Management

In real operations, constantly engaging or being heavily loaded in task execution would increase the risk exposure and reduce the capability of task handling for agents, which might cause unexpected deficiencies in operations. Training with the proposed HTLM Dec-POMDP framework and the properly designed reward functions allows agents to make intelligent decisions with their preferences and to be prepared for additional/unexpected task loads. In this study, we investigate the impact of varying levels of idle incentive and reassignment penalty in reward functions on team performance and load reduction. We assume that agents have distinct capability levels in firefighting and rescue as shown in Table 3.2, capabilities do not change throughout the operation, and agents use the same decision model and reward function. As described in Section 3.2.2, for R_{II}, agents receive a positive reward when idling and a negative reward if the capability level is 2-level higher than the believed severity level of the assigned task, reflecting the penalty on unnecessary assignment. For the task change penalty T_{RP}, agents receive a negative reward if being reassigned to a task at a different location. While the reward functions can be adjusted by agents to model their preferences in operations, this study focuses on the change in the reward magnitude given to the team instead of the difference in reward functions among the agents.

Table 3.2: Capabilities for a Heterogeneous Team

Agent ID	Firefighting	Rescue
1	3	1
2	3	2
3	2	0
4	0	1
5	1	1

We train the team of agents with 6 different combinations of idle incentive R_{II} and reassignment penalty T_{RP}, and evaluate the trained agents with two performance metrics as shown in Fig. 3.2 with and without R_{II} under (a) no, (b) medium, and (c) high reassignment penalty T_{RP}. The agents are evaluated in 500 operation trials, and each operation has 30 steps. The first performance metric is operation effectiveness in terms of the steps taken to complete all tasks. When

training with no and medium T_{RP}, team effectiveness remains at the same level and is not affected by the idling behaviors induced by idle incentive. When T_{RP} is too high, agents are less willing to move and an additional idle incentive further decreases the effectiveness. The second metric is the unused capability per agent per step. Whenever an agent decides to idle, the capability of the agent is considered as unused. On average, each agent has a total task-handling capability level of 2.8 (1.8 in firefighting and 1 in rescue) as indicated in Table 3.2, and most of the capability $\approx 71\%$ is saved throughout an operation compared to $\approx 2.5\%$ when there are no idle incentives as shown in Fig. 3.2. The amount of unused capability without R_{II} is not always 0 because of the randomness and agent's exploratory behavior during training.

Next, we showcase the impact of reassignment penalty T_{RP} on load management in Fig. 3.3 by fixing the amplitude of idle incentive. Giving more reassignment penalties during training incentivizes agents to reduce reassignment or overall traversing costs if the penalty is the distance between two task locations. Agents are trained using different levels of reassignment penalty (no, medium, and high), and in evaluation, three metrics are investigated: (a) operation effectiveness, (b) team idle count before completion, and (c) team reassignment count before completion (Fig. 3.3). For training with medium T_{RP} levels, the team ends up with more idling behaviors and less frequent reassignment before completing all tasks without compromising the operation effectiveness compared to no T_{RP} penalty case (Fig. 3.3(a)). When T_{RP} is too high during training, agents prefer to stay at the same task or stay idling, but operation effectiveness cannot be guaranteed.

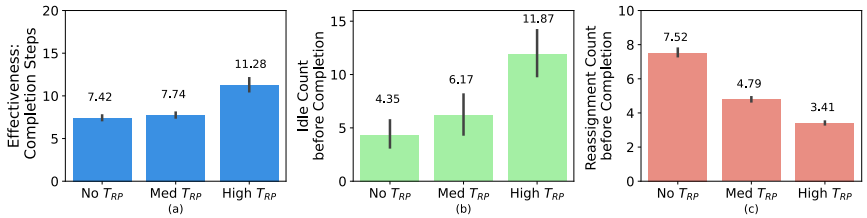
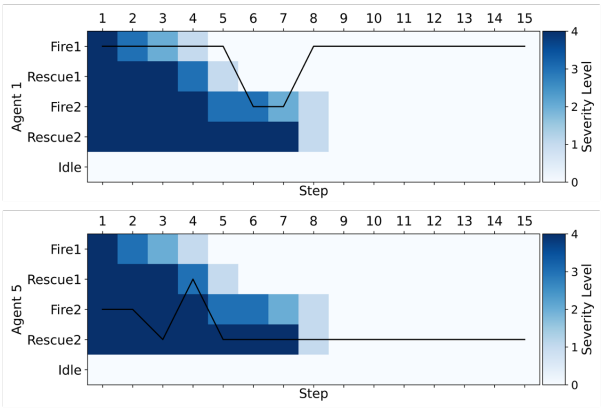
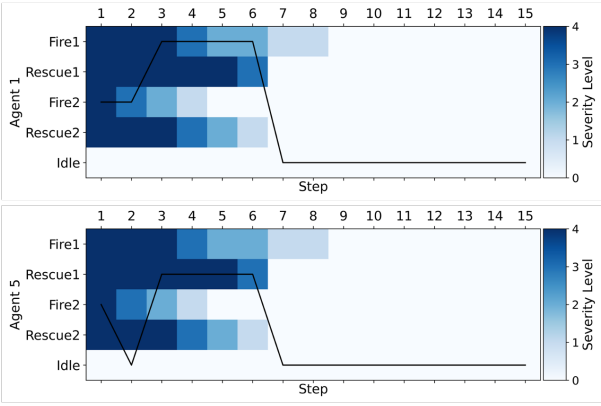


Figure 3.3: Effect of reassignment penalty T_{RP} on operation effectiveness (a) and load conservation (b), (c). Bar plots show the mean and the standard error for each performance metric evaluated using 500 realizations.



(a) Decisions of Agent 1 and 5 trained with no R_{II} and no T_{RP}



(b) Decisions of Agent 1 and 5 trained with R_{II} and medium T_{RP}

Figure 3.4: Demonstration of task allocation in an operation, where agents are trained (a) with no R_{II} and no T_{RP}, and (b) with R_{II} and medium T_{RP}. Black lines indicate agent decisions for the first 15 steps, and color blocks represent severity levels for each task ranging from 0 to 4.

To visualize the learned agent behaviors in our proposed HTLM Dec-POMDP, we demonstrate the intelligent task allocation decisions made by two agents in the team shown in Fig. 3.4b and compare to the decisions trained with no idle incentive or reassignment penalty shown in Fig. 3.4a. With the help of idle incentive and reassignment penalty, agent task load is reduced by idling when its capability is not needed and when all tasks are completed. Agent 1 with a firefighting capability of 3 gets assigned to fire tasks when fire levels are above 1 and idles when fire level is 1 (steps 7 and 8). At steps 2, 7, and 8, Agent 5 with both firefighting and rescue capability of 1 stays idle, letting other agents work on the remaining tasks. Furthermore, the reassignment count for both Agent 1

and 5 is only 1 (step 3 for Agent 1 and Agent 5 in Fig. 3.4b), while the reassignment count is 2 for both agents trained with no T_{RP} (steps 6, 8 for Agent 1 and step 4, 5 for Agent 5 in Fig. 3.4b).

3.3.2 Agent Importance

Before training a team of agents for an operation, it is complicated to decide which agents should be chosen as team members. Random team composition based on agent capabilities might cost additional training time if the selected agents are not capable enough to accomplish tasks efficiently or cost too much capability usage if they are all being deployed. The proposed HTLM Dec-POMDP framework could provide insights into team formation and agent importance in terms of their trained behaviors and available capabilities. Identifying the importance of agents could help in selecting team members and inferring team resilience when one of the trained agents is lost or not performing well in the team. A metric for agent importance depending on capability usage and task urgency is proposed in the following equations:

$$\zeta = \sum_{k=1}^{K} \omega_k u_k, \tag{3.9}$$

$$u_k = \frac{1}{E} \sum_{i=1}^{E} c_k \mathbf{1}(\pi^* = a_k | i), \tag{3.10}$$

$$\bar{\omega}_k = \sum_{i=1}^{E} \sum_{j=1}^{N} \mathbf{1}(\pi_j^* = a_k | i), \tag{3.11}$$

$$\omega_k = \frac{\bar{\omega}_k}{\sum_{k=1}^{K} \bar{\omega}_k}. \tag{3.12}$$

Here, the importance metric $\zeta := f(U, W | C, \pi^*, n_s)$ for each agent is defined as a function of its capability usage on each task type $U = [u_1, \ldots, u_K]$ depending on its available task-specific capabilities $C = [c_1, \ldots, c_K]$ and the task urgency weight $W = [\omega_1, \ldots, \omega_K]$, where K is the number of task types in an operation. The importance is estimated through E simulated operation

episodes and the trained optimal agent policy π^* (i.e. task allocation decisions). The capability usage for each task type $u_k \in U$ is computed as the average capability used in E episodes as shown in Eq. (3.10), where a_k indicates the agent is allocated to task type $k \in 1, \ldots, K$. However, tasks have different dynamics, and some tasks are harder to resolve than others. Taking our fire and rescue operation as an example, for each step, fire severity can only be reduced by one level at most, and multiple people can be rescued only when the fire level is low, which naturally makes firefighting tasks harder and more urgent. Capability usage only accounts for the resource needed for each task, but an important agent would spend more capability on more urgent tasks. Instead of making arbitrary weights for task urgency, we assume that the task allocated more frequently is more urgent. With a team of N agents and E episodes, the estimated task urgency weight is computed as Eq. (3.11) and normalized using Eq. (3.12). Then, the importance of each agent is calculated using Eq. (3.9).

Table 3.3: Agent Importance for Three Teams

Agent	C_{fire}	C_{rescue}	U_{fire}	U_{rescue}	ζ
1	3	1	3.44	0	**2.58**
2	3	1	22.38	0	16.75
3	1	1	10.80	0	8.08
4	2	2	3.70	4.26	3.84
5	2	2	8.99	9.42	9.10

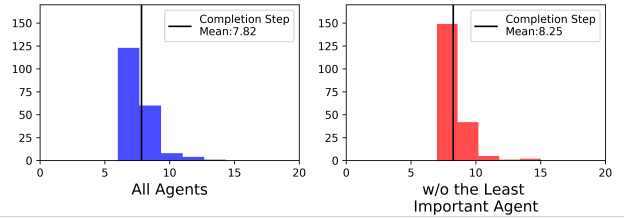
(a) Semi-Heterogeneous Team $\omega_{fire} = 0.75, \omega_{rescue} = 0.25$

Agent	C_{fire}	C_{rescue}	U_{fire}	U_{rescue}	ζ
1	2	1	17.91	1.49	12.49
2	2	1	12.68	4.05	9.84
3	2	1	2.52	3.20	**2.74**
4	2	1	12.17	3.00	9.15
5	2	1	14.60	2.99	10.77

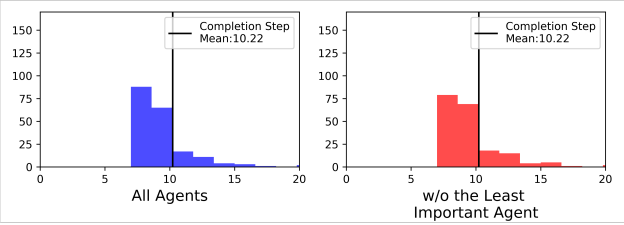
(b) Homogeneous Team $\omega_{fire} = 0.67, \omega_{rescue} = 0.33$

In this analysis, we use the same operation scenario with $h = 30$ steps, consisting of two task types (fire and rescue) at two locations and five agents. The importance score ranges from 0 if the agent always idles to $c_k h$ if all agents always work on the same task type. In addition to a team

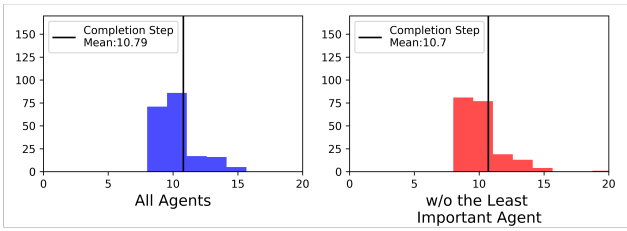
of five agents with heterogeneous capabilities as described in Table 3.2, a semi-heterogeneous team and a homogeneous team are considered. The available capabilities (C_{fire}, C_{rescue}), used capabilities (U_{fire}, U_{rescue}), and the importance ζ of each agent for three teams are summarized in Table 3.3. The agent with the least importance score for each team is highlighted in bold.



(a) Heterogeneous Team



(b) Semi-Heterogeneous Team



(c) Homogeneous Team

Figure 3.5: Operation effectiveness comparison between having all agents (blue) and inactivating the least important agent (red) for (a) Heterogeneous Team, (b) Semi-Heterogeneous Team, (c) Homogeneous Team. Each histogram shows 200 realizations with the mean completion step indicated.

To validate the agent importance, without re-training the agents, we set one of the agents in a team to always idle and evaluate the operation effectiveness described as a completion step compared to having all agents. Fig. 3.5 suggests that the operation effectiveness is not affected after

inactivating the least important agent indicated in Table 3.3 for all three teams. Inactivating the most important agent for all teams would lead to operation failure (i.e. unable to accomplish tasks within 30 steps). It is worth noting that the agent importance metric is not simply the indicator of its capability but depends also on the learned decision process during the training in interaction with other agents. Consequently, agents with the same capability might have different importance (Table 3.3b), and agents with higher capabilities might be the least important and unnecessary to execute any tasks (Table 3.3a). Hence, the agent importance is the metric for the significance of the agent in a learned team collaboration. We have demonstrated that the importance metric for agents trained with the HTLM Dec-POMDP framework could help to infer which agent is more significant in collaboration, how much capability could be saved without affecting the operation effectiveness, and consequently how much additional demand can be handled by the team by fully activating all agents.

3.4 Conclusions

In this work, we formulate an intelligent decision-making framework for multi-agent teams to learn efficient task allocation strategies with load management through deep reinforcement learning. The proposed framework (HTLM Dec-POMDP) models idling as an option for allocation decisions and encourages idling behaviors to conserve task load and available resources by customizing individual reward functions based on agent preferences. Training multi-agent teams with the HTLM Dec-POMDP framework allows agents to reduce task load without compromising the operation effectiveness and provides the opportunity to investigate team member selection, team resilience, and precaution preparation. Results show that properly defined reward functions can achieve load reduction and resource conservation while maintaining team performance. A metric for agent importance is defined to measure agent significance within the team and to suggest team formation and reliability.

The proposed method serves as an incentive for the development of intelligent multi-agent task allocation systems with the consideration of load management, which could be beneficial for

designing management-level strategies in complex operations for human-autonomy teams. The method could potentially suggest decisions to human agents in real practices without being overloaded. In large-scale problems, as more tasks and agents are involved, a logistic of subteam formation and a hierarchical task planning method is desired to enhance computational complexity. Moreover, the study of agent importance demonstrates asymmetric behaviors in collaboration with respect to capabilities, which potentially opens the discussion on studying collaborative teaming behaviors in the future.

CHAPTER 4

Human-Autonomy Teaming in Immersive Environments

This chapter leverages the efforts of learning coordination strategies discussed in Chapter 2 and Chapter 3 to prepare autonomous agents for teaming up with actual human decision-makers in a simulated disaster relief operation to evaluate the learned expertise.

4.1 Introduction

Autonomous agents are increasingly thought of as team members alongside humans in large-scale operations. Both autonomous agents and humans have inherent advantages and limitations that might affect the outcome of an operation. The inclusion of autonomy within a team requires a significant effort to prepare autonomous agents to interact with human decision-makers. A fundamental challenge emerging is how to train humans and autonomous agents to achieve the operation goals with maximum performance. In this chapter, we develop a game-based platform to train and quantitatively analyze the performance of human-autonomy teams in a disaster relief scenario. This platform allows a human decision-maker to team up with autonomous agents in a high-fidelity immersive environment to accomplish a mission with cognitive workload monitoring. A team of autonomous agents with various capabilities is trained using reinforcement learning to interact with the human decision-maker. A machine learning-based virtual adaptive user interface (AUI) is developed to assist humans in situation awareness and decision-making by measuring the cognitive task load and providing team capability information at a suitable level of complexity. We quantitatively investigated the behaviors and the performance of the trained human-autonomy

teams and compared various proposed AUIs. The developed game-based platform allows to study coordination and communication of human and autonomous agents in reproducible time-critical operations, providing opportunities to conduct comprehensive studies including edge cases and unforeseen events. The immersive environment is a fundamental modeling and simulation tool for further development and design of human-autonomy teams.

Multi-agent systems offer opportunities to generate strategies for operations involving complex interactions between autonomy and humans. Human-autonomy teaming [7], as an emerging research area, is beneficial when team members are collaborative and trustworthy, and they achieve superior effectiveness by completing tasks that they cannot accomplish individually. The inclusion of humans in autonomous systems is challenging in aspects of safety, computational and cognitive overload, reproducibility for training, data collection, and edge case modeling, especially for large-scale applications including military operations [109], such as humanitarian relief and battlefields. With the advancement in artificial intelligence (AI), autonomy is shifting the role from a subordinate member to a team member who shares the equal responsibility of planning, reasoning, and execution as a human agent. Research questions regarding human factors, autonomy training, and team resilience then arise in the U.S. Army Ground Vehicle Systems Center (GVSC), Army Research Laboratory, etc. to address the benefits and challenges in human-autonomy teaming.

Extensive research has been focusing on human-machine collaboration, trust and communication, cognitive workload, learning from demonstration, teleoperation, and team design, most of which require data-driven methods to improve performance and ensure application transferability. However, the limitations and challenges in collecting large amounts of data from human subjects and in studying a variety of case scenarios motivate further developments in realistic and reproducible simulations. Recent advances in high-fidelity simulations and immersive environments using extended reality [110] have led to a paradigm shift in training and validating human-autonomy teaming systems.

The aim of this chapter is to develop a game-based platform to allow human decision-makers to interact and collaborate with autonomous agents in a simulated disaster relief scenario created in an

immersive environment. Each autonomous agent is equipped with an AI decision-making process trained to dynamically allocate tasks in response to operational demands. Humans can perceive the environment by assessing the local situation and receiving information through a virtual adaptive user interface (AUI). By monitoring human cognitive workload, the AUI is trained to provide a suitable level of information complexity to assist decision-making and situation awareness. We develop different machine learning-based architectures for the AUI and analyze the training performance. The platform empowers future studies in learning management-level strategies, analyzing emergent collaborative behavior, and adjusting to individual preferences for human-autonomy teams.

4.1.1 Human-Autonomy Teaming

In recent decades, vast improvements in multi-agent task allocation have been made in homogeneous team compositions such as swarm robots, and the solution methods often rely on optimization with the consideration of temporal and spatial constraints [11]. Heterogenous teaming in complex operations with task specialization, collaborative behaviors, and risk aversion has also been studied to improve the overall team performance [27]. Autonomy offers functional efficiency and the capability of handling hazardous environments but has limitations in reaction to unforeseen situations. At the same time, humans are adaptive and creative in resolving sudden events but may suffer cognitive overload in busy working environments. Combining the advantages of humans and autonomy has encouraged research in human-autonomy collaboration including shared decision-making [111] and workload-adaptive support in task allocation [112]. Although many task planning and scheduling problems can handle constraints between tasks and agents, more complex operations require strategies to satisfy dynamically changing demands due to uncertainties and adversaries. In addition, decentralization of multi-agent task allocation provides more robust and scalable solutions, compared to a centralized system, especially when encountering the failure of one or some of the team members. Therefore, there is a push towards human-autonomy teaming, considering more complex environments and diverse teams, by having autonomous agents as team

members alongside humans (Figure 4.1).

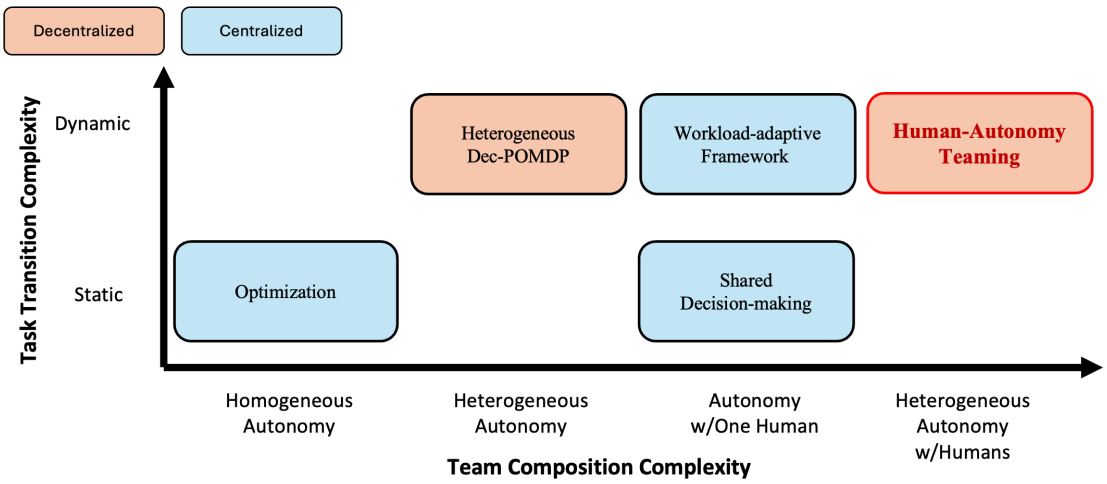


Figure 4.1: Methods of multi-agent task allocation in literature by team composition and task dynamics.

4.1.2 Cognitive Workload

In complex and large-scale operations, humans play significant roles in processing information, decision-making, supervising, and operating. Humans are capable of handling and adapting to unforeseen events but may suffer from fatigue over time due to workload and stress. Survey approaches including NASA TLX and SAGAT [113] are used when physiological data is too invasive. Physiological data are useful for monitoring human cognitive workload and stress. Researchers have used neuroimaging techniques including electroencephalography (EEG) and functional near-infrared spectroscopy (fNIRS) to monitor brain activities for mental workload assessment [114–116]. Wearing these sensors might not be practical in real operations especially when involving drastic movements. Other physiological metrics such as heart rate variability (HRV), photoplethysmography (PPG), electrodermal activity (EDA), and galvanic skin response (GSR) are available on wrist-wearable technologies and are extensively studied in the literature [117]. Given increasing computational power, deep learning techniques have become prevalent [118], particularly in driver awareness assessment [119, 120], by using an eye-tracking device that cap-

tures images and measures pupil dilation. However, most studies involving multiple subjects reveal limited effectiveness of physiological metrics as predictors of cognitive workload [121, 122], because human subjects have more complex interactions (knowledge, expertise, memories) with the environment and autonomy that cannot be measured by physiological sensors and are the actual cause of the cognitive workload.

4.1.3 Adaptive User Interface

There are multiple ways to utilize the information extracted from measuring cognitive workload and assessing operational demands. To improve human-machine collaboration by allowing autonomy to be aware of humans, one can embed trust and cognitive workload-based models for dynamic task allocation in human-machine teams [123]. Work also has been done to automate decision-making by adaptively commanding unmanned vehicles [124] and dynamic role assignment [125]. Studies on the adaptive user interface (AUI) have shown improved usability and user performance [126] and increased efficiency in human-autonomy teams [127] by providing adaptive decision suggestions. As high-fidelity simulations become a useful platform for human-autonomy collaboration research, further investigation on providing selective content and extracting important information in virtual and immersive environments is needed.

4.2 Methods

4.2.1 Human-Autonomy Teaming Architecture

A game-based platform is developed to enable the interaction between humans and trained autonomous agents in a virtual immersive environment, as shown in Figure 4.2. A complex operational scenario is simulated in a high-fidelity environment using a game engine where a human decision-maker interacts with the environment in real-time using a virtual reality headset and controllers. Autonomous agents possessing various capabilities are trained using a decentralized artificial intelligence algorithm to allocate tasks and collaborate with team members through rein-

forcement learning. All agents observe local task demands depending on their locations and then make a task allocation decision based on the received information and past experience. Physiological data including pupil dilation, heart rate, and sweat gland activity are measured from the human during the time of the operation. This data is input to a trained machine learning architecture to correlate cognitive workload and situation awareness of the human during the operation. To monitor the cognitive load of the human and to improve the overall performance during operation, a virtual adaptive user interface is constructed as a virtual tablet in the game engine to provide information at a suitable level of complexity to the human in decision-making depending on the status of the cognitive load. In this section, we will illustrate the details of the immersive environment, how the autonomous agents are trained, and how the human is interfaced with the environment.

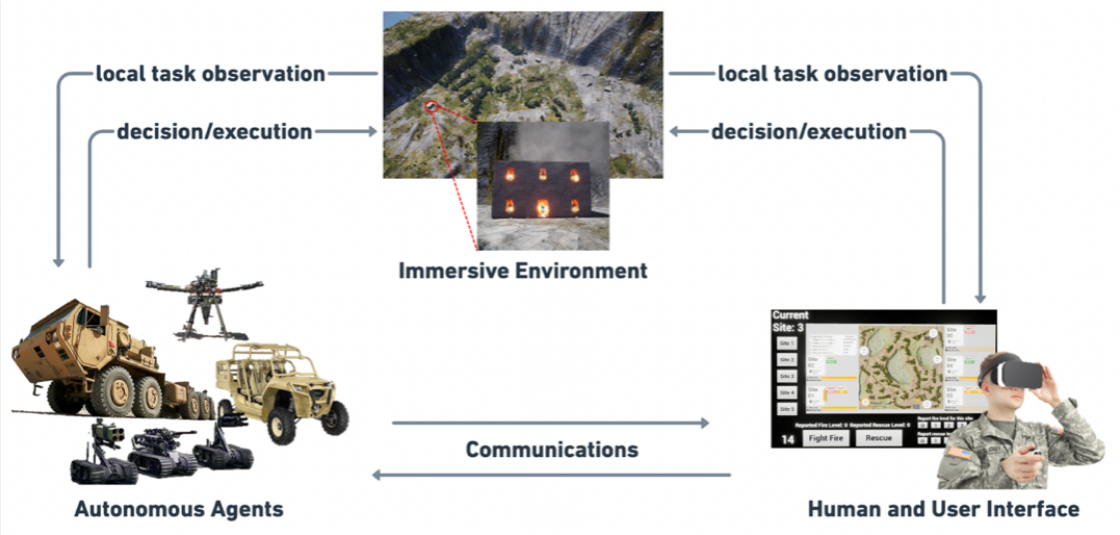


Figure 4.2: A game-based platform for humans to interact with autonomous agents in an immersive environment using virtual reality. Humans access information from both the environment and the virtual user interface.

4.2.1.1 Immersive Environments

To analyze the teaming behaviors and human-autonomy collaboration, a disaster relief operation scenario is developed in the Unreal Engine, where a VIVE Pro Eye virtual reality headset and game motion controllers are connected to allow an immersive experience for humans. The disas-

ter relief operation involves five disaster sites, a team of six autonomous agents, and one human agent with the same attribute set but different capabilities. At each site, there are two conditionally dependent tasks (i.e., fire-extinguishing and rescuing people; people cannot be rescued if the fire level at that site is high) with various demand levels within the range of 0 to 4 and stochastic task dynamics. The operation is considered a success if the demand levels of all 10 tasks are reduced from 4 to 0 (i.e., no fire or people to rescue anywhere) within 20 time steps, at which each agent can choose to execute a task or idle. The attribute set for each agent includes two task-specific capabilities: firefighting and rescue. The capabilities are added and become joint capabilities if agents are assigned to the same task. The task-specific joint capability level is ranging from level 1 to 5. The higher the joint capability is, the higher the probability the task demand would reduce. The challenging aspects of this disaster relief operation come from not only the stochasticity and complexity in task dynamics but also the partial observability of the agents and the decentralization of the decisions. Agents can neither observe the full status of the environment nor know the decisions of the others beforehand. Agents are allowed to communicate the task demands they locally observed, but the information accuracy is not guaranteed. The inaccurate information represents the uncertainties in agent sensing capabilities and information exchange.

4.2.1.2 Artificial Intelligence Decision-making Framework

The task allocation strategy is formulated as a Heterogeneous Teaming with Load Management Decentralized Partially Observable Markov Decision Process (HTLM Dec-POMDP). HTLM Dec-POMDP [128] discussed in Chapter 3 Section 3.2. It is defined as a tuple $\langle S, \{A_i\}, T, R, \{Z_i\}, O, \gamma \rangle$, where i is the agent index and $s \in S$ is the actual state of the environment represented as the task demand levels and the joint capability levels of the agents. The overview of the decision-making process is demonstrated in Figure 4.3. Instead of having a centralized system allocate tasks for the team, at time step t, each agent makes its own decision in a decentralized manner and executes action $a_i \in A_i$ based on the current information. Then, the environment state s transitions to the next state s' with probability $T = \Pr(s'|s, a)$. Agents receive reward $r = R(s, a)$ and partial ob-

servation $z_i \in Z_i$ with probability $O = \Pr(z|s', a)$, and communicate their observed task demand levels with each other to make decisions in the next step. Agents make synchronized decisions to eventually reach the goal. A higher reward is given if task demands are low, and the reward is discounted by $\gamma \in [0, 1]$ at each step to incentivize faster operation completion. In the HTLM Dec-POMDP formulation, the heterogeneity is characterized by the multiple types of tasks and the variety in agents and agent task-specific capability levels. To manage agent task load, agents are given the option to idle to avoid excessive capability usage and unnecessary task reassignment.

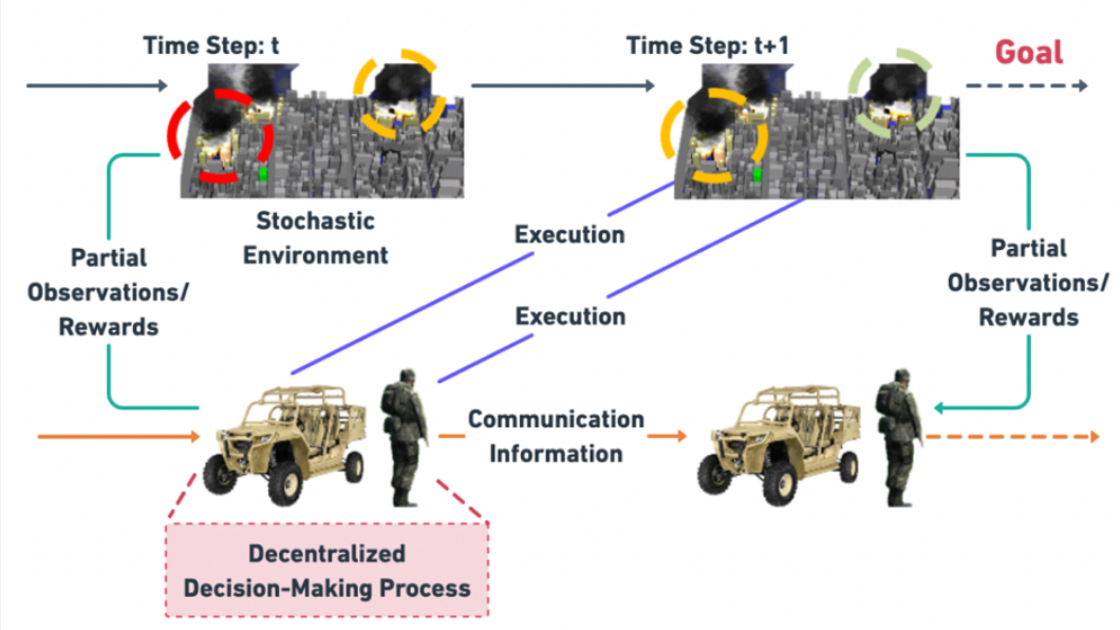


Figure 4.3: Decentralized Partially Observable Markov Decision Process. At time step t, each agent makes a decision and executes the action. Then, the environment transitions to the next state with uncertainties. Agents receive rewards and local observation and communicate observed task levels with each other to make decisions in the next.

4.2.1.3 Training Autonomous Agents

In the multi-agent task allocation problem setting, it is nearly impossible to train autonomous agents alongside real human decision-makers due to the amount of data required, especially for large and complex operation scenarios. Therefore, with the help of simulation reproductivity and reinforcement learning techniques, we treat the human agent as one of the autonomous agents in

the team and let autonomous agents interact with this artificial human agent (AHA) in the simulated environment to efficiently produce data for learning task allocation strategies.

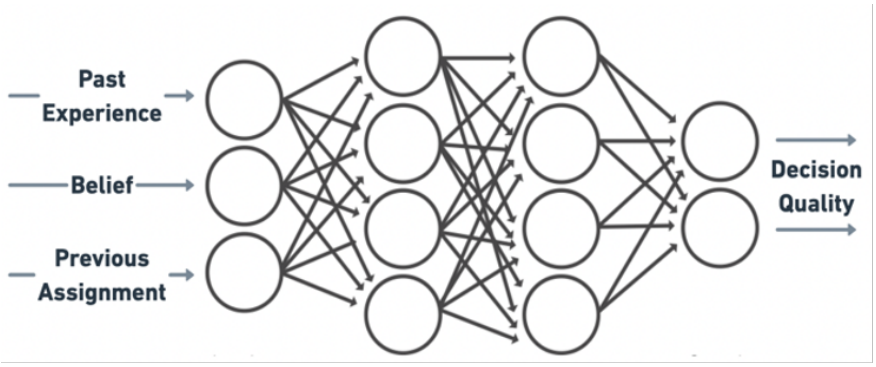


Figure 4.4: Deep Q-network with load management.

Instead of reasoning on all the possible observation-action-communication histories given the uncertainties, agents make decisions based on a summary of the histories. Deep Q-learning with Load Management [128] utilizes the concept of belief, the probability state-space over the environment state, to update the understanding of the environment state at every step. Each agent is equipped with a deep Q-network $Q(K, a; \theta)$ parametrized by θ to make optimal task allocation decisions as shown in Figure 4.4. The network inputs the knowledge state K that includes the belief summarizing the agent's past experience and the agent's previous assignment to account for task reassignment activities. The trained network estimates the optimal quality of each decision $Q^*(K, a)$ defined under Bellman Optimality as equation 4.1. The optimal task allocation policy π^* can be simply retrieved by equation 4.2. Given enough transition data $\langle K, a, K', R(K, a)\rangle$, the Q-network converges to a target Q-value $y = R(K, a) + \gamma \max Q(K', a'; \psi)$, where the target Q-network $Q(K', a'; \psi)$ parametrized by ψ is updated less frequently. During training, the loss between the Q-network and the target Q-value defined in equation 4.3 is minimized.

$$Q^*(K, a) = R(K, a) + \gamma \max Q^*(K', a') \tag{4.1}$$

$$\pi^*(K) = \arg\max Q^*(K, a) \tag{4.2}$$

$$L = \mathbb{E}\left[y(K', a'; \psi) - Q(K, a; \theta)\right]^2 \tag{4.3}$$

The artificial human agent (AHA) is trained together with autonomous agents to ensure collaborative teaming behaviors. To better prepare autonomy to collaborate with actual human decision-makers, autonomous agents need to know how to account for human factors. Hence, we incorporate the AHA with a human characteristic described in [27]: humans make noisy rational decisions [129] in real operations, which might be beneficial in the presence of unexpected events. Essentially, using the output of the Q-network, the AHA can take any action with the defined probabilities in equation 4.4, where $\phi \in [0, \text{inf})$ is a tuning parameter determining how rational the agent is (0: random; inf: rational). To take advantage of the trained AHA, we let this AHA run in the background during operations to provide decision suggestions and help analyze the decision quality of actual human decision-makers.

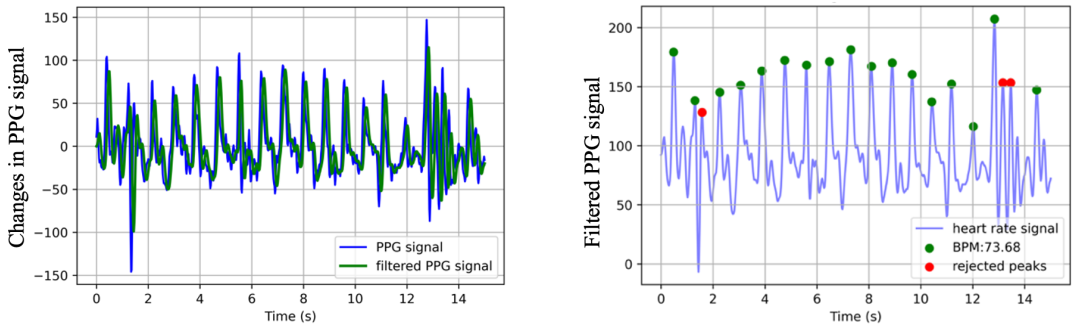
$$\Pr(a \mid K) = \frac{\exp(\phi \cdot Q(K, a))}{\sum_{a_i} \exp(\phi \cdot Q(K, a_i))} \tag{4.4}$$

4.2.2 Interfacing for Human Agents

4.2.2.1 Workload Measuring and Processing

Throughout the operation, humans would experience different levels of task demands and digest different levels of information complexity. Monitoring the cognitive workload is essential to help understand the stress status of the human and potentially improve the teaming performance. Three types of physiological data are measured from the human during the operation to estimate the cognitive workload: (1) pupil dilation – the pupil size in diameter is measured using VIVE Pro Eye VR headset; (2) heart rate – photoplethysmography (PPG) is measured by Shimmer3 GSR+ device (shimmer, n.d.) to monitor heart rate; (3) galvanic skin response (GSR) – sweat gland activity is also measured by Shimmer3 GSR+ as an indication of emotional arousal.

To utilize workload measures for providing a suitable level of information complexity, each of the above raw physiological data is processed and categorized into Low/Medium/High using the participant's thresholds to estimate the workload for each at each decision time step. For each decision interval, we compute the average pupil dilation and the average GSR, and the raw PPG



(a) Filtering PPG signal

(b) Converting filtered PPG to BPM

Figure 4.5: PPG signal is recorded during a decision interval of 15 seconds, filtered (a), and converted to BPM (b).

signal is filtered and converted to beats per minute (BPM) using heartPy [130] as shown in Figure 4.5.

4.2.2.2 Local Environment and Virtual User Interface

During the operation, the human agent can only observe the local environment including task demands of the local disaster site and autonomous agents at the same site as shown in the left two pictures of Figure 4.6. The human agent also has access to a virtual user interface (Figure 4.6, right) controlled by two VIVE Pro Eye motion controllers. The left controller controls the position of the virtual user interface and can turn the site information on and off. The right controller allows the human agent to make selections. The virtual user interface includes the following components:

- Center – a 2D map of the operation environment and received information regarding task demand levels at each disaster site indicated by the orange bar, agent location at the previous step, and agent capabilities indicated by green, yellow, and red color (high, medium, and low),
- Top left – current site location of the human agent,
- Bottom left – time in seconds remaining for making a decision,
- Yellow boxes – selections of site locations, tasks, and observations to report,

- Blue box – suggested decisions from the trained artificial human agent.



Figure 4.6: Demonstration of the local environment (left) and the virtual adaptive user interface (right) accessible to the human agent. AUI Center – site information; top left – current site location; top bottom – decision count-down; yellow boxes – selection area for locations, tasks, and observations; blue box – suggested decisions.

4.2.2.3 Cognitive Task Load Adaptive User Interface (CTL-AUI)

Given the urgency of the time-critical operation scenario, the human agent is supposed to gather information from the local disaster site and from the user interface, make a task allocation decision, and report observed task demands within a limited time frame. Detailed information displayed by the interface provides thorough environment status but might lead to longer processing time, added workload, as well as not delivering decisions in time. Therefore, under different environment demand levels and workload levels of the human agent, providing a suitable complexity level of information will help maintain a reasonable workload and improve the overall team performance.

Figure 4.7 shows three designed interface choices with different complexity levels of information. Low complexity (Figure 4.7a) only shows where the human agent was in the previous time step; medium complexity (Figure 4.7b) shows estimated sub-team quality at each disaster site measured by the addition of agent capabilities; and high complexity (Figure 4.7c) details each agent's capability at every site. At medium and high complexities, team quality and agent capabilities

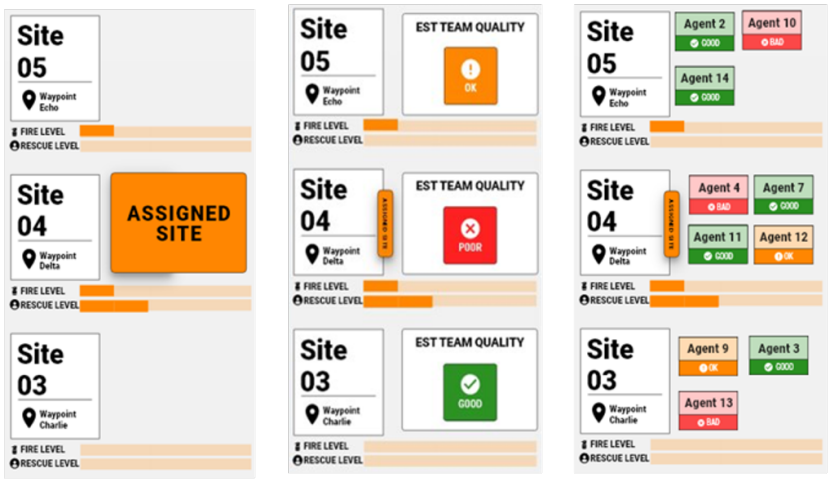


Figure 4.7: Three interface choices providing either low, medium, or high information complexity level (left to right).

are expressed with color coordination to capability quality at that site. Green is representative of a good quality rating, meaning the capability at that site exceeds demand; orange shows an okay rating, meaning there is enough capability to meet demand; Red is a bad rating, indicating there is more demand at the site than the capability can meet.

We propose an intelligent cognitive task load index adaptive user interface (CTLX-AUI) framework as shown in Figure 4.8 to determine the best information complexity provided to the human agent based on varying workload conditions (WL) and environment demands (EnvD). The red lines indicate that the data are only used during training. The framework aims to investigate the approaches of correlating the WL and EnvD to the task load index and therefore improve the design of adaptive UI.

In the framework, all data are categorized into three levels (low, medium, or high). To ensure the algorithm can be applied to different humans, level thresholds of the workload measures of each participant are identified for categorization. The first block is the cognitive task load (CTL) classifier, which estimates the situation awareness measured by the task load index (TLX) using WL and EnvD. The ground truth TLX score data used for training is collected from multiple participants by answering a simplified NASA TLX Questionnaire during the operation as shown in Figure 4.9.

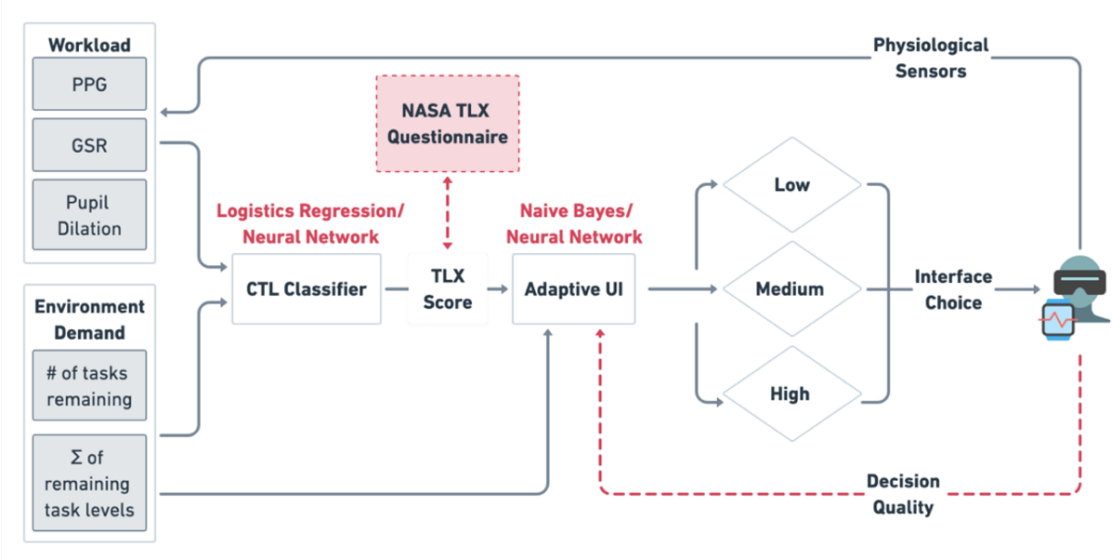


Figure 4.8: Training and operation of cognitive task load index adaptive user interface (CTLX-AUI) by estimating NASA TLX score. Red lines indicate the data is only used for training the interface.

The TLX score is categorized into low, medium, and high levels. We try two approaches for the CTL classifier: (1) logistic regression, a linear classifier with polynomial nonlinear features, to maintain interpretability, and (2) neural network. The second block is the adaptive user interface (AUI) classifier, which determines the interface complexity level to provide given the estimated TLX score, EnvD, and the resultant decision quality in low/medium/high. Two approaches for the AUI include Naïve Bayes and neural network. Both estimate the interface choice using TLX score, EnvD, and decision quality. In operation, the questionnaire is not needed, the CTL classifier estimates the TLX score for the human participant using WL and EnvD measures, and AUI predicts the best interface choice by setting the desired decision quality to be high.

While the CTLX-AUI framework explicitly measures the task load index to help understand situation awareness, such an approach might lead to increased bias and simplified data features. Therefore, we propose another architecture CTL-AUI, shown in Figure 4.10, which directly predicts interface choice using a neural network given all WL, EnvD, and decision quality data. Two proposed architectures will be studied and compared in the next section.

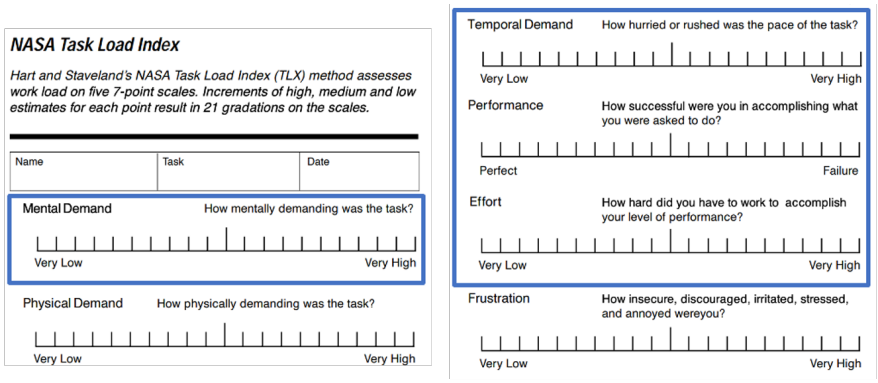


Figure 4.9: Simplified NASA TLX Questionnaire.

4.3 Experiment and Results

4.3.1 Experiment Setup

4.3.1.1 Disaster Relief Operation

In the operation, there are 5 disaster sites, and each site has 2 different but conditionally dependent tasks: extinguishing the fire and rescuing victims (a total of 10 tasks). The demand level for each task is ranging from 0 to 4. The human-autonomy team consists of six autonomous agents, and one artificial human agent with various levels of task-specific capability set: [extinguishing, rescuing], and the perception ability of the agents are fixed to provide high observation and information accuracies (80%). The capability level of each agent is ranging from 0 to 4, and the joint capability level when agents cooperate at the same task is computed by adding the capabilities of the agents assigned to the same task and being capped at level 5. The fire task level reduces by 1 with higher probabilities when the joint extinguishing capability level is high and the rescuing task level reduces by the amount of the joint rescuing capability level deterministically. All task levels would not increase once they reach level 0. In addition, the dependency between tasks at the same site is considered in dynamic state transition probabilities. Agents cannot rescue people until the fire level at that site is below 3. The reward function is designed so that agents would not waste high capability levels on low-level tasks, and they would reach the completion state (all

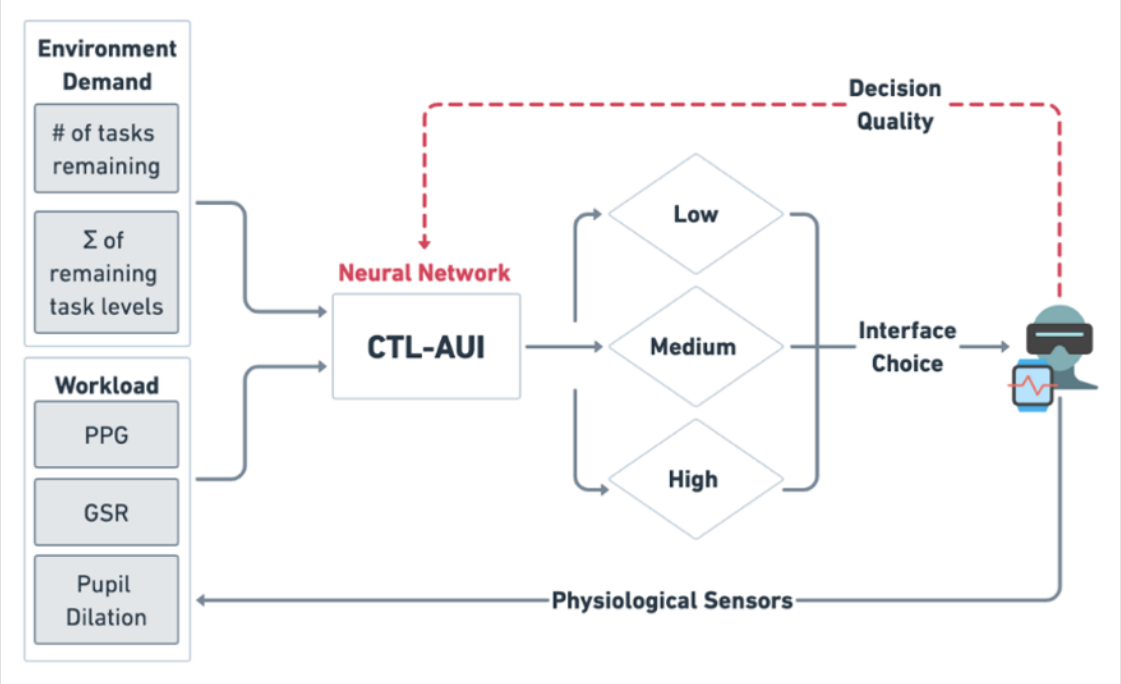


Figure 4.10: Training and operation of cognitive task load adaptive user interface (CTL-AUI) by direct classification. Red lines indicate the data is only used for training the interface.

task demand levels = 0) with fewer steps. The artificial human agent is given ≈10% chances of not making rational decisions and will be replaced by a human participant in further studies. This team is trained using deep Q-learning for load management proposed in HTLM Dec-POMDP [128].

4.3.1.2 Team Behavior Visualization

We demonstrate the decision behaviors of a 7-agent team in a 5-site fire and rescue operation in Figure 4.11. Agents collaborate on high-demand fire tasks (i.e., step 1), split to handle different low-demand tasks and idle when no actions are necessary. On average of 200 realizations of the operation, the team manages to complete the tasks with 11.16 steps and a 95% success rate. However, to highlight the impact of a human decision-maker, removing the artificial human agent would lead to a 4% success rate and 14.75 average steps for successful completion. The trained autonomous agents require significant help from the human agent for effective teaming, and the trained policy is not adaptive to cover the missing contribution of the human agent.

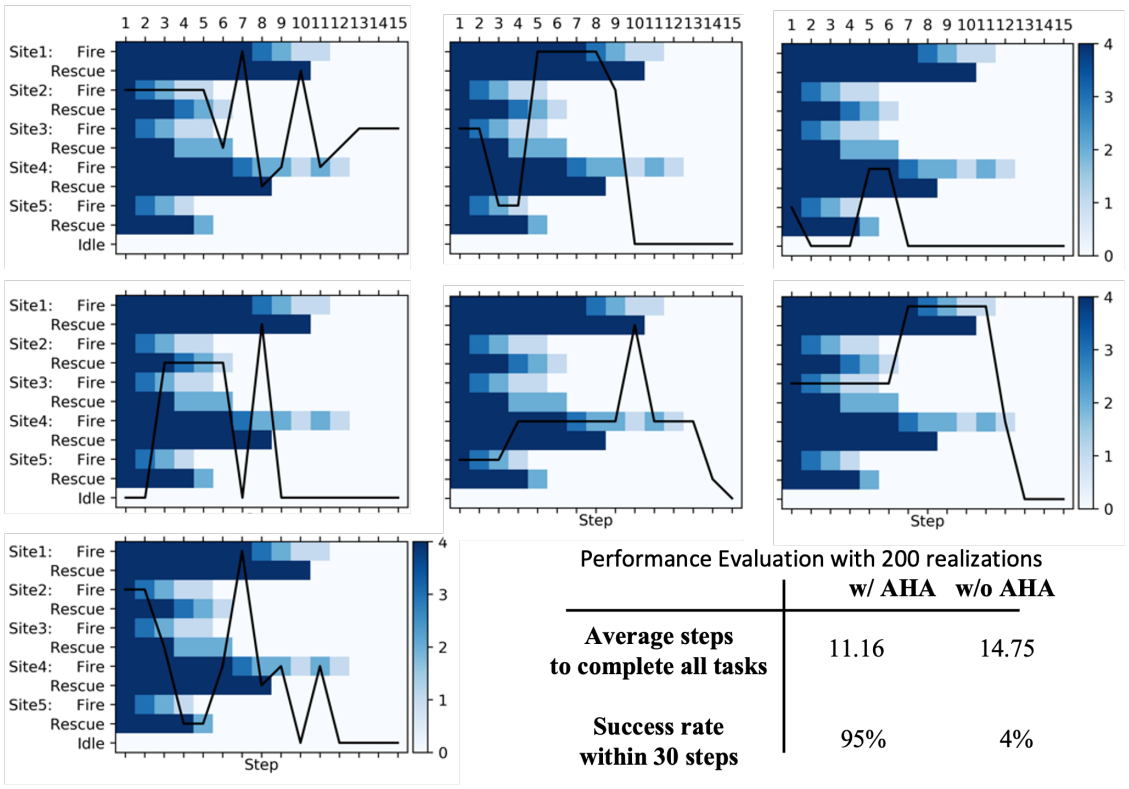


Figure 4.11: The behaviors of a team of six autonomous agents (top 6) and one artificial human agent (bottom) on a 5-site fire and rescue operations. The team performance is evaluated with and without AHA to highlight the impact of human decision-makers.

4.3.1.3 AUI Training with Human Participants

To train the adaptive user interface (AUI), we collect behavioral data from human participants playing with the trained autonomous agents as a team. We disable the suggestions provided by the artificial human agent (AHA) since it provides additional information that may lessen the impact of the information complexity. The disaster relief operation will start with all 10 tasks at level 4 and last 20 steps. For each step, the participant will have 15 seconds to make a task allocation decision and 10 seconds to answer the TLX questionnaire (post-assessment for each decision step), while all the workload metrics and environment demands are recorded. The data we collect include six participants playing the same operation over four different random seeds (i.e., the probabilities of task transition and the observations may vary). After removing the data after they complete all tasks, there are a total of 271 data samples used for training AUI.

To decide whether the decision made by the participants is a high/medium/low-quality decision, we use two approaches to evaluate: 1. By the AHA trained with autonomous agents – AHA contains Q-values for all actions given a state, and the decisions with top-ranked Q-values are considered high-quality decisions. 2. By a game expert – By replaying the decisions made by each participant, the expert is given enough time and information to evaluate if the decisions are rational and therefore are high-quality decisions.

4.3.1.4 Accuracy of Cognitive Task Load Adaptive User Interface

We first investigate the training of the Cognitive Task Load (CTL) Classifier that summarizes environment information and workload measures to estimate the NASA TLX score. Data samples are split into training and testing sets (8:2). Two training methods, logistic regression with polynomial features (LR-CTL) and neural network (NN-CTL), are tested. The accuracies, computed as the average correct prediction rate on the test dataset over 30 training instances, for LR-CTL and NN-CTL are 60.18% and 58.73% respectively. Although the accuracies seem not high, it is challenging in workload assessment when multiple subjects are involved [115], where an accuracy of 43% is reported by using only physiological sensors as opposed to 65% using brain activity signals (EEG+fNIRS). Different participants may have different experience levels with the immersive environments and hold different opinions on the TLX questionnaire. There are factors other than physiological activities and environmental information to be captured to have a better estimation.

Table 4.1: Test Accuracy of CTL-AUI Variants

AUI Architectures	**CTL Classifier**	**AUI Classifier**	**Decision Quality Evaluated by**	**Average Accuracy**	**Max Accuracy**
LRNB-CTLX-AUI	Logistic Regression	Naïve Bayes	AHA	44.28	46.13
NN-CTLX-AUI	Neural Network	Neural Network	AHA	44.35	48.17
CTL-AUI-AHA	Neural Network		AHA	**49.52**	**56.46**
CTL-AUI-Expert	Neural Network		Game Expert	48.12	52.77

Next, we test the performance of the two proposed AUI architectures with several variants with the average and maximum accuracy over 30 training instances, as summarized in Table 1. The architectures that directly estimate the interface choice trained by a neural network (CTL-AUI)

demonstrate higher prediction accuracies than the ones that have an additional estimation step for the NASA TLX score. Directly using environment information and workload measures provides better accuracy, but it is hard to explain the situation awareness or workload status using this neural network approach. Since the decision quality given an interface choice is measured by the AHA and a game expert, it does not reflect all participants' opinions on the interface choice. There is a significant improvement compared to using random interface choice (expected 33%).

4.4 Human Subject Study

The main goal of this study is to analyze the collaboration of human and autonomous agents in a game-based platform developed for human-autonomy teaming in a virtual environment. This study is planned to involve numerous human participants to record their decisions and physiological responses while playing the disaster relief operation game in a variety of designed scenarios and use the data to quantitatively analyze the effect of human factors and cognitive load on the effectiveness of the human-autonomy teaming operation. Specifically, the research objectives of the study include:

(a) Demonstrate benefits and limitations of human participants in teams with autonomous systems under different levels of task load;

(b) Measure human's cognitive load while playing the game with simulated autonomous systems by multiple physiological parameters such as pupil dilation, photoplethysmography (PPG) that monitors heart rate, and galvanic skin response (GSR);

(c) Test an adaptive video and audio interface in which the information complexity provided to the human, during the game is optimized to reduce cognitive load;

(d) Investigate the effect of providing decision recommendations on team performance and human decision-making.

The study is approved by IRB: Health Sciences and Behavioral Sciences, University of Michigan, Ann Arbor, MI, USA (Study ID: Study ID: HUM00207942).

4.4.1 Operation Scenarios

A virtual reality disaster relief game is created, where there are a number of houses on fire and people trapped in houses. The decisions include which location to go to and which task to choose (fighting the fire or rescuing people) at each step of the game so that all the fire tasks are extinguished and all the people are rescued as soon as possible. The simulated autonomous agents (AI teammates) automatically play the game and decide what action to take based on the trained decision-making process and the state of the game (e.g. severity of fire at each building, number of people trapped in each building). The human participant wears a virtual reality headset and works as a team member in collaboration with autonomous agents to extinguish fires and rescue people. The participant uses game controllers to make decisions that would affect the state of the operation. The game is performed in discrete time steps. At each step of the game, the participant can only choose one action which can be fighting fire or rescuing people at one of the houses. This is where human reasoning and decision-making process play an important role because humans should evaluate the environment and identify which action is the best one to take so that the overall disaster situation improves faster. Then, the participant will observe the outcome of the decisions made by the team (human participant and autonomous agents) and decide what action to take in the next step. Throughout the operation, pupil dilation is measured by the VR headset, and galvanic skin response (GSR) data and photoplethysmography (PPG) data that monitor heart rate are measured by the shimmer device. These physiological data are then used by the adaptive user interface (AUI) to provide the most suitable level of information complexity. The AUI additionally provides the human participant with information regarding the overall game status (fire level, rescue demand, and number of agents at other locations) and can suggest to humans what decision to make that the human might or might not agree with provided by the artificial human agent.

The participants played 6 games (scenarios) in this study as indicated below. Each scenario has

20 decision steps to complete, and the participants are given 15 seconds to make their decisions.

- Scenario 1 Constant interface with the least complex information; without suggested decisions

- Scenario 2 Constant interface with the most complex information; without suggested decisions

- Scenario 3 Interface with adaptive information complexity; without suggested decisions

- Scenario 4 Interface with adaptive information complexity; with suggested decisions

- Scenario 5 Interface with adaptive information complexity; with suggested decisions in an unknown sudden event — the fire level suddenly increases to the maximum level 4 at location 2 at step 8.

- Scenario 6 Interface with adaptive information complexity; with suggested decisions with decreased decision time (10 seconds).

4.4.2 Data Collection

50 human subjects have participated in the study, and the participants are screened based on the following criteria:

- Age: Be at least 18 years and at most 65 years old;
- No disorders related to visual/hearing/motion;
- Have no known disorders or injuries that may affect the ability to perform game controllers or play in a simulated environment for up to 90 minutes.

4.4.2.1 Physiological and Behavioral Data

For each participant and each operation scenario, we collect the following physiological data, following the same process in Section 4.2.2.1:

- Pupil dilation; average value for each game decision step

- Heart rate beat per minute (BPM); converted from PPG using heartPy for each game decision step
- GSR; average value for each game decision step

and the game operation data for each decision step:

- Operation state; the remaining demand level for each task
- Interface complexity level
- Decisions of all team members
- Belief distributions of task levels for each autonomous agent including the artificial human agent
- Q-Values of all autonomous agents including the artificial human agent

4.4.2.2 Questionnaire Data

The pre-session questionnaire includes:

- What is your experience of using Virtual Reality? (1:none, 5: familiar)
- What is your experience of the game environment? (1:none, 5: familiar)
- How comfortable do you expect to work/collaborate with simulated agents? (1:not comfortable, 5:comfortable)

The post-session questionnaire includes:

- How comfortable were you working with simulated agents? (1:not comfortable, 5:comfortable)
- How frequently did you pay attention to other simulated agents? (1:no attention, 5: always)

- How often did you make random decisions? (1:always logical decision, 5: always random decisions)
- How often did you rely on recommended decisions? (1:always make my own, 5: always follow recommended decisions)

4.4.3 Exploratory Analysis Results

The exploratory data on pre-/post- questionnaires and performance in operation scenarios are summarized in Figure 4.12 and Figure 4.13 respectively.

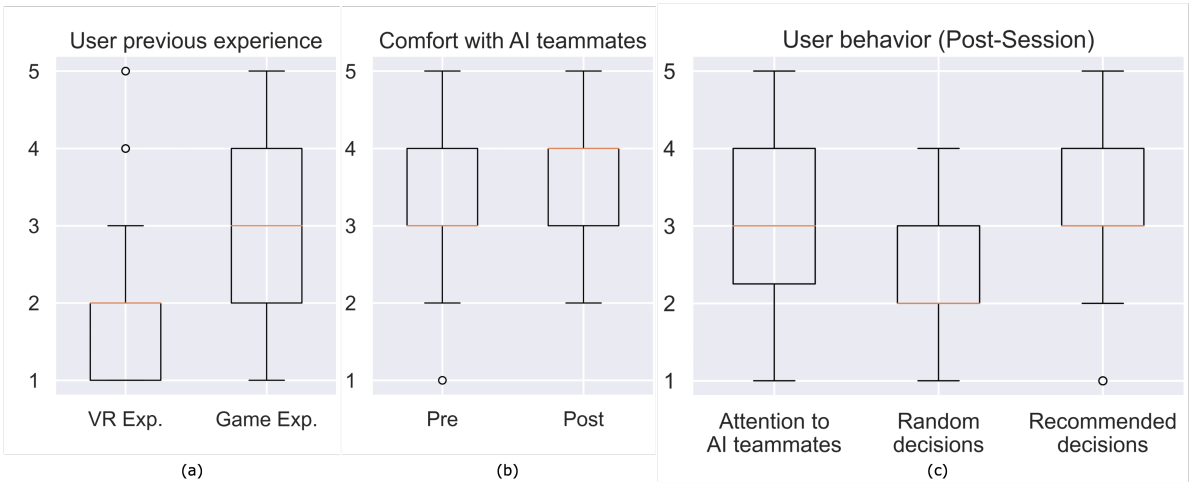


Figure 4.12: Participant responses to pre- and post- questionnaires. (a) Their past experience with VR and game environments. (b) Their opinions on how comfortable working with AI teammates. (c) Their attention to AI teammates and decision processes.

With the questionnaire responses, the results show that 1) there is an increase in how comfortable the participants feel about working together with AI teammates as an indicator of trust; 2) utilizing the information on AI teammates may not be the priority in decentralized decision-making; 3) participants rarely make completely random decisions and do not always follow recommendations. These observations help further investigate how the teaming performance varies with different participant profiles.

In operation scenarios, the number of steps to complete all tasks is used as the performance metric. The game is considered a failure when participants cannot finish tasks in 20 steps. The

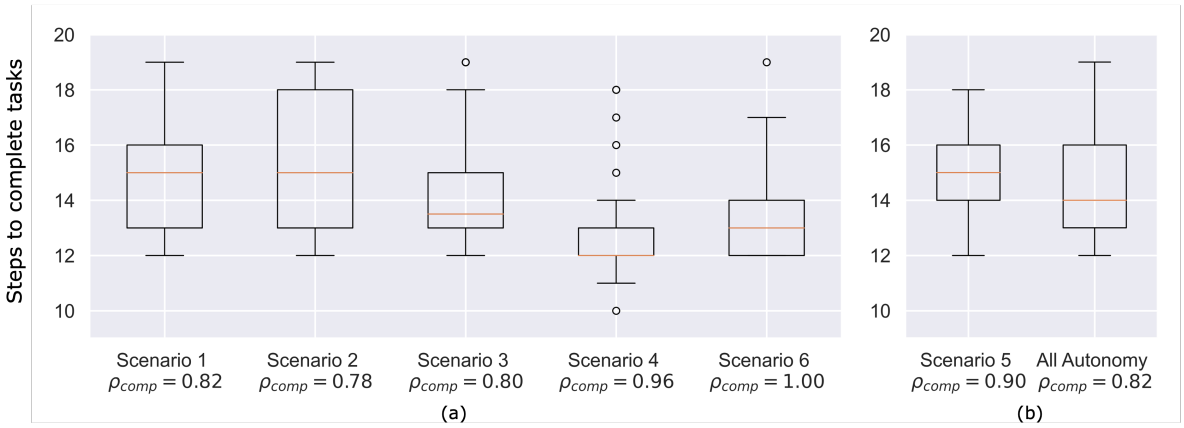


Figure 4.13: Participant performance (steps to complete all tasks) in operation scenarios. (a) Without the sudden event. (b) With the sudden event, compared to the all-autonomy team.

success rate for each scenario is denoted as ρ_{comp}. When the adaptive user interface is first introduced in Scenario 3 without recommendations, the performance shows improvement (1.5 steps faster in median) compared to the constant information complexity (Scenario 1 and 2). Also, when the recommendations are provided in Scenario 4, the performance and the success rate further improve. When the sudden event (increase in the fire level at location 2 at step 8) is introduced in Scenario 5, the success rate (90%) is higher than the all-autonomy team (82%). In Scenario 6, as the decision-making time is compressed to 10 seconds per operation step, participants tend to either be more focused or rely on recommendations, maintaining a high success rate and performance.

4.5 Conclusions

As autonomy becomes more intelligent, instead of playing assistant roles in human-autonomy collaboration, autonomy has shared authority in strategic planning and task execution. Human-autonomy teaming requires significant efforts in understanding human characteristics and decision strategies as well as generating data in reproducible and edge scenarios. In this chapter, we demonstrate human-autonomy teaming in a disaster-relief operation, where the strategies of autonomous agents are trained using deep reinforcement learning, and a human player is interfaced with physiological sensors and provided with a virtual adaptive user interface to summarize communicated

information from other agents. The developed CTL-AUI helps further investigate the impact of human situation awareness and adaptation on human-autonomy teaming behaviors, compared to constant UIs. The simulation platform also helps study how wrong information and unexpected situations that lead to cognitive overload can be mitigated. Although substituting an artificial human agent with a real human decision-maker allows the team to successfully complete the mission, humans are not always making the trained rational decisions and the desired synchronized collaboration is not optimal (i.e., humans can never outperform the recommended decisions in sufficiently trained scenarios). The advantages of having human agents are their creativity in reacting to sudden events and special knowledge in unforeseen situations. Additional efforts are needed to verify whether autonomy can raise situation awareness, understand human strategies, and realize evolving situations from human-demonstrated decisions for better collaboration and teaming performance. Our work in developing a platform for humans to team up with autonomy in immersive environments bridges the reality and the virtual. By allowing humans to interact with customizable autonomy assets in reproducible scenarios, this study provides insight into preparing autonomy to collaborate with human decision-makers, understanding human strategies in teaming settings, and therefore achieving superior effectiveness of human-autonomy teaming in large-scale operations.

CHAPTER 5

Evaluating Emergent Coordination in Multi-Agent Task Allocation through Causal Inference and Sub-Team Identification

With the decision-making framework developed in Chapter 2 and multi-objective behavior learning proposed in Chapter 3, it remains unclear how and why certain decisions are made under different operation demands. Such understanding becomes especially challenging when collaborating with humans as discussed in 4. This chapter presents post-hoc interpretation techniques that allow autonomous teammates to understand the teaming behavior trained using the methods proposed in the learning phase and the behavior demonstrated by human teammates.

5.1 Introduction

In real-world multi-agent teaming scenarios, coordinated strategies are naturally formed and driven by a common goal and individual preferences, without explicit communication of intended actions. Advanced technologies in collaborative autonomy have significantly enhanced the effectiveness of multi-agent teams including swarm robots and human-machine teams. Understanding complex interactions, behavior correlations, causality, and evolving strategies of multi-agent teams remains challenging. Coordination in multi-agent task allocation often considers centralized consensus among distributed systems through game theory and optimization [131] or utilizes deep reinforcement learning techniques [132] to train policies under uncertainties by interacting

with the environment. To investigate multi-agent coordination strategies, inverse reinforcement learning techniques [97, 133, 134] enable learning from human demonstrations, and explainable reinforcement learning aims to create interpretable reward functions for agent behaviors. There is a need, however, to quantify the synchronization and synergy of teaming behaviors under different operations to gain insights into the underlying team dynamics and the emergent coordination strategies.

Multi-agent reinforcement learning (MARL) allows agents to learn coordinated behaviors, where team strategies evolve from random exploratory to objective-oriented exploitative behaviors [27, 128]. In cooperative and competitive tasks, rewards are used to find the correlations between agent behaviors [48], and population-based methods [135] select team members by competition. Sometimes high-performing MARL-trained teams in predator-prey systems do not reach the desired level of collaboration [49], especially in large teams [136]. Although correlation in rewards and performance metrics help illustrate the evolving behaviors during learning, the coordination patterns depicted by agent decisions have not been studied, and whether a suitable strategy is implemented facing unexpected situations is therefore not guaranteed. Studying coordination allows analyzing and determining team strategies in various situations by directly using operation demands and agent behaviors.

To analyze decisions made by multi-agent teams and the cause-and-effect relationships among these decisions, it is important to visualize and measure whether agent decision patterns can be informed by past decisions of other agents, and whether certain agents form sub-teams that are highly synchronized compared to others. In this chapter, we utilize the concepts of decision causality [43] and bidirectional weighted networks [96] for evaluating coordination in decentralized multi-agent teams operating in large-scale task allocation problems. Measuring the level of team coordination and identifying highly coordinated sub-teams are crucial for understanding emergent strategies and recognizing the need for them in evolving situations and unforeseen events, especially when agents are facing extreme workload and stress, and when agents over- or under-estimate the task demand. We demonstrate the developed metrics for quantifying team coordination in multi-agent

systems operating in an environment with predefined centralized rule-based strategies and decentralized team strategies trained by MARL. We also show that RL-trained decentralized multi-agent teams can learn not only effective operation completion but also coordinated behaviors, compared to rule-based policies.

The key contributions of this work can be summarized as:

- Developing a generalized approach to measure causality among agents and identify coordinated sub-teams using time series data of agents' decisions in decentralized multi-agent teams,
- Defining a quantitative metric for emergent coordination to acquire patterns in rule-based and reward-driven strategies across various environment demand levels,
- Investigating the effect of over- and under-estimating task demands by agents on team coordination and performance.

To our knowledge, this work is the first attempt to apply causal inference for evaluating coordination behaviors in multi-agent task allocation, which is capable of quantitatively investigating agent-level coordination, sub-team formation, and team-level strategies.

Training agents to learn policies for coordination and competition has shown promising effectiveness using MARL [48, 128, 135]. Studies have involved the construction of coordination graphs [137] and distributed networks [138] using deep learning techniques [139] to learn the quality values for the joint action of each agent pair, and demonstrated the benefit of using correlation in reward histories on learning team strategies [48]. Such learning methods utilize the underlying coordination structure to guide the learning process, but they have not focused on analyzing the learned strategy or coordination level across different teams or situations.

To address the issue of black box decisions made by artificial intelligence, explainable reinforcement learning [34–36] tends to evaluate the interestingness elements in agent decisions such as frequency, execution uncertainty, and favorable/adverse situations [37]. Several studies have investigated learned formation strategies [50] and path planning strategies using heat maps [51].

However, these approaches have not explained the correlation or causality effect between agents, and they are not general enough to merely use state and decision information. Moreover, it would become challenging to study task allocation problems when multiple strategies lead to optimal team performance.

The Pearson correlation [38] and KL divergence [39] are widely used to measure the relationship between two random variables, to match the probability of the desired strategy [40], or to identify strategy switching [41], but these metrics produce a symmetric correlation between variables and do not address the fact that correlation does not imply causation. In economics and ecosystems, statistical tests including Granger Causality [42] and Convergence Cross Mapping (CCM) [43] have been developed to detect the causality between time series of two variables. While Granger Causality assumes that variables are separable, CCM variants [44–47] seek synergistic effects between variables in stochastic and dynamical systems. Causality has been rarely studied in multi-agent task allocation to quantitatively measure the level of coordination in strategies.

The novelty of our approach includes the application of the asymmetric causation technique in task allocation and the extension from two-agent causation to quantification of multi-agent coordination by identifying synergistic sub-teams using community finding techniques [140].

5.2 Methods

In the context of multi-task multi-agent decision-making problems, the team coordination operation is considered as a complex simulated scenario where tasks, represented by demand levels, are to be accomplished in multiple time steps by agents equipped with various task-handling capability levels. The operation is accomplished once the demand levels of all tasks are reduced to zero. With the presence of uncertainties, the dynamics of the task demand levels are stochastic and dependent on each other and/or the agents' capability levels at the assigned tasks. This flexible formulation of the environment is applicable in various types of scenarios ranging from homogeneous-task homogeneous-agent with one-step task assignment (i.e., predator swarm robots) to heterogeneous-

task heterogeneous-agent with multi-step dynamic task allocation (i.e., human-autonomy teaming). In such an environment, agents are incentivized to accomplish tasks within fewer time steps. The coordination strategy is emerging from cooperating on the same task to simultaneously working on different tasks for efficient teaming. Quantitative analysis of team coordination is necessary for understanding emergent teaming behaviors and varying strategies under changing situations. Therefore, we propose a method that utilizes bidirectional causality in decisions and coordinated sub-team identification to evaluate team coordination.

5.2.1 Preliminaries

Formulation: Dec-POMDP

The environment is formulated as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [20], which is defined as a tuple $< S, \{A_i\}, \mathcal{T}, R, \{Z_i\}, \mathcal{O}, \gamma >$, where i is the agent index and $s \in S$ is the actual state of the environment. In the context of task allocation, states are represented as task demand levels. At each time step, each agent i receives partial observation $z_i \in Z_i$ (i.e., the demand level of the assigned task), and the joint observation is $z = \{z_i\}$. Then, each agent i makes a task allocation decision and executes action $a_i \in A_i$, and the joint action is $a = \{a_i\}$. With actions taken, the current environment state s transitions to the next state s' with probability $\mathcal{T} = Pr(s'|s, a)$. Agents receive reward $r = f_R(s, a)$ and joint observation $z = \{z_i\}$ with probability $\mathcal{O} = Pr(z|s', a)$ for making decisions in the next step. In an episode of h time steps, agents follow policy π to make decisions. The cumulative discounted reward is $R_c = \sum_{t=1}^{h} \gamma^{t-1} f_R(s^t, a^t)$, where $\gamma \in [0, 1)$ is a discount factor introduced to provide more reward at earlier time steps.

Decision Strategy: Reinforcement Learning vs. Rules

By interacting with the environment, agents associate their past experience with the reward they received. RL-trained methods allow agents to learn the optimal policy that maximizes the cumulative discounted reward R_c, but the learned behaviors are often hard to explain [37]. In contrast, rule-based decisions are comparably interpretable. For example, the rule could be simply form-

ing groups of three to cooperate in addressing highly demanding tasks and individually executing different low-demand tasks. However, these strategies may not always be optimal or effective.

Convergence Cross Mapping as Coordination

When two agents express coordinated behaviors, the behavior of one agent is predictable from the others. Thus, two agents are coordinating not only when they work on the same task, but also when the next decision of one agent is predictable by observing the past decisions of the other agent. Such predictability is considered as the causal influence one time series variable has on another, i.e., bidirectional causation, which can be examined by Convergence Cross Mapping (CCM) [43]. CCM is a causation measure first used for variables in ecosystems. CCM reconstructs the state space for each variable using D-dimensional embedding of τ-delayed coordinates (i.e., the shadow manifold M) by assuming that there are shadow variables driving the underlying dynamics of the observed variable. When two variables are from the same dynamical state space (manifold), they are causally coupled. Using CCM, when two agents X, Y (with shadow manifolds M_X, M_Y) are perfectly coordinated, the points nearby M_X will match points nearby on M_Y for every time step in the manifolds. CCM then uses a simplex projection [46] technique to measure causality by finding the nearest $D + 1$ neighbors of M_X and the distances to those neighbors as weights to compute a cross-mapping estimation in M_Y.

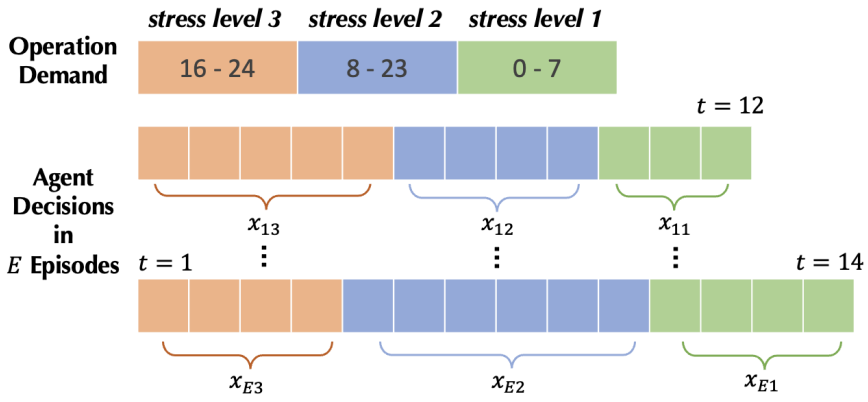


Figure 5.1: Demonstration of aggregating short decision time series in $N = 3$ stress levels across E simulated episodes, for an operation with $p = 6$ tasks and $l = 4$ maximum task demand level.

5.2.2 Coordination in Different Stress Levels

Stress Level

When agents in a team experience different levels of demand (stress), the agents may behave differently. The stress levels can be determined by the demand or urgency levels of an operation, computation load, or cognitive load for human agents. We aim to study the emergent coordination between agents as a team by analyzing agent decision behaviors at different stress levels. In our task allocation problem formulation, the operation environment state is represented by a number p of tasks with maximum task demand level l, and the total demand level ranges from 0 to pl. We categorize stress levels based on the total demand level. For example, three equally divided demand level ranges $[0, pl/3)$, $[pl/3, 2pl/3)$, $[2pl/3, pl]$ can represent low, medium, and high-stress levels respectively.

Coordination between Two Agents

CCM requires a long time series for convergence. In multi-agent task allocation, the decision time series are often short (i.e., less than 30 sequential decisions), depending on the number of agents, tasks, and task demand levels. Multi-Spatial Convergence Cross Mapping (MSCCM) [47] utilizes dewdrop regression [141] to compute CCM prediction skills with repeated observed experiences. Given the reproducibility of a simulated environment with uncertainties, multiple episodes of short time series, sharing similar underlying dynamics, can be aggregated to create long time series. Therefore, we adapt MSCCM to discrete multi-agent Task Allocation CCM (TA-CCM) to quantify coordination at different stress levels by measuring the correspondence between the reconstructed state spaces (shadow manifolds) of two aggregated decision time series.

Let $i \in \{1, \ldots, E\}$ be the index of E simulated episodes of agents completing tasks, $j \in \{1, \ldots, N\}$ be the index of N stress levels, and let $x_{ij},\ y_{ij}$ denote the time series of task allocation decisions for agent x and agent y at stress level j in episode i. As demonstrated in Fig. 5.1, the aggregated decision time series for each stress level j are $X = [x_{1j}, \ldots, x_{Ej}],\ Y = [y_{1j}, \ldots, y_{Ej}]$. Since each agent's decisions are segmented by stress level in the same way for each episode, X and Y have the same length. The shadow manifolds $M_X,\ M_Y$ are constructed

with lag τ and embedding dimension D, where $M_X(t) = [X_{(t)}, X_{(t-\tau)}, \ldots, X_{(t-(D-1)\tau)}]$ and $M_Y(t) = [Y_{(t)}, Y_{(t-\tau)}, \ldots, Y_{(t-(D-1)\tau)}]$. The simplex projection technique estimates the decisions of both agents using the other's shadow manifold for each time step t, denoted as $\hat{X}(t)|M_Y$ and $\hat{Y}(t)|M_X$, and the prediction skills are computed using the Pearson correlation coefficient ρ between the estimated and observed decisions, i.e. $\rho(\hat{X}|M_Y, X)$ and $\rho(\hat{Y}|M_X, Y)$ respectively. Such high prediction skills in an agent pair infer high coordination of the two agents.

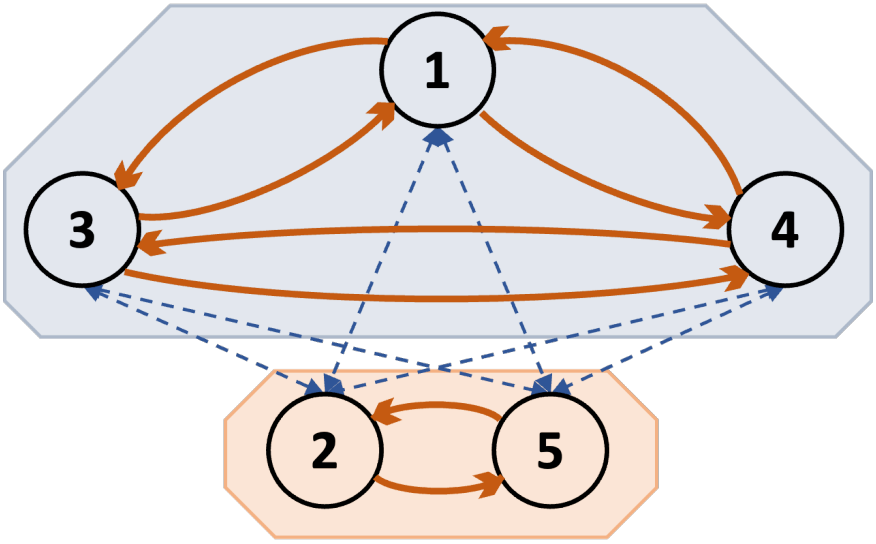


Figure 5.2: A weighted directed network of a team of five agents with edges represented by behavior prediction skills. Solid orange arrows indicate strong predictability given an agent, dashed blue bidirectional arrows indicate weak predictability between agents, and shades indicate the two sub-teams.

5.2.3 Sub-Team Identification

With the help of TA-CCM described in Section 5.2.2, the coordination in any agent pair can be measured using bidirectional prediction skills. Note that the complexity to compute the team coordination is $O(N^2 - N)$, and it can be reduced only if agents are known to be deterministic and predictable. One simple way of quantifying team-level coordination is to sum up the prediction skills between all pairs of agents. This approach requires high prediction skills between all agents to achieve high team-level coordination. One reasonable assumption is that the team has high coordination as long as every agent is highly coordinated with at least one other agent, and therefore

multiple coordinated sub-teams with low prediction skills across different sub-teams can have high team-level coordination. For example, as illustrated in Fig. 5.2, a team of 5 agents split into two sub-teams with strong predictability skills within the sub-teams can still be considered highly coordinated. Therefore, we construct a weighted directed network with vertices being agents and edges being TA-CCM prediction skills, and adapt a community finding technique: Modularity Maximization (ModMax) [96] for directed networks to identify statistically significant sub-teams. The number of sub-teams and the sub-team sizes do not need to be specified using ModMax.

Let $\mathcal{P}$ be edges of a directed network or an n by n matrix constructed by the prediction skills measured by TA-CCM, where n is the team size and $\mathcal{P}_{ij} = \rho(\hat{Y}_j|M_{X_i}, Y_j)$ denote the correlation coefficient between predicted decisions of agent j given agent i and observed decisions of agent j. Let κ_j^{in}, κ_i^{out} denote the in- and out-degrees of the agents, indicating how predictable agent j is and how useful agent i is to predict others respectively. We have $m = \sum_{ij} \mathcal{P}_{ij} = \sum_j \kappa_j^{in} = \sum_i \kappa_i^{out}$. ModMax allows us to find a division of a network with high benefit modularity U defined as:

$$U = \frac{1}{m} \sum_{ij} \left[\mathcal{P}_{ij} - \frac{\kappa_j^{in} \kappa_i^{out}}{m} \right] \delta_{g_i, g_j}, \tag{5.1}$$

where δ is the Kronecker delta, and g_i is the sub-team to which agent i is assigned. Large positive modularity U indicates statistically higher coordination within the sub-teams than expected. This process outputs a set of sub-teams $\{g\}$ that maximizes U given the prediction skill matrix $\mathcal{P}$.

5.2.4 Team Coordination

The sub-team identification process described in Section 5.2.3 tends to find sub-teams in which agents are more coordinated than those in different sub-teams. Therefore, we neglect the coordination across other sub-teams, and the team coordination only depends on the coordination levels and the sizes of the sub-teams. The team coordination score $\mathcal{I}$ can then be computed as:

$$\mathcal{I} = \sum_{\forall g} \omega_g C_g, \tag{5.2}$$

$$C_g = \frac{1}{|g|^2 - |g|} \sum_{\forall i \neq j} \mathcal{P}_{ij} \delta_{g_i, g_j}, \tag{5.3}$$

$$\omega_g = \frac{|g|}{n}, \tag{5.4}$$

where $\{g\}$ is the set of identified sub-teams, C_g is the sub-team coordination score, and ω_g is the sub-team size weight with respect to the full team size n. The sub-team coordination score C_g is the average prediction skills between different agents within the same sub-team. $|g|$ denotes the size of the sub-team, and $|g|^2 - |g|$ is the total number of directional prediction skills with self-prediction excluded. The team coordination metric is in the range of $[0, 1]$. Given a prediction skill matrix with off-diagonal elements being 0 (i.e., everyone is unpredictable), the team would have 0 coordination score. When all sub-teams are perfectly coordinated ($C_g = 1$), the team coordination score will sum up to 1. The advantage of this metric based on sub-team identification and weighted by sub-team size is that it provides a generalized measure to all types of teams with different team sizes. Note that we can introduce a hyperparameter ψ to the sub-team size weight (i.e., $w_g = (|g|/n)^{\psi}$) to favor the coordinated sub-team with larger team size. Here, we choose $\psi = 1$ by assuming each coordinated sub-team contributes equally to the team coordination.

5.3 Results

To analyze the emergent team coordination in multi-agent task allocation, we consider a Heterogeneous Teaming Decentralized Partially Observable Markov Decision Process (HT Dec-POMDP) [27] formulation, where the dynamics between task demand levels and agent capability levels are explicitly modeled, agents have their own decentralized decision models based on the beliefs over tasks, and each agent updates its belief by observing local demand and sharing its own observation and is rewarded upon the completion of tasks. Although the HT Dec-POMDP allows modeling heterogeneous tasks and agents where agents may specialize in certain tasks, it is more challenging to identify how a certain behavior emerges when tasks and agents are homogeneous, i.e., a symmetry-breaking phenomenon [142]. In this study, we consider a number of indexed

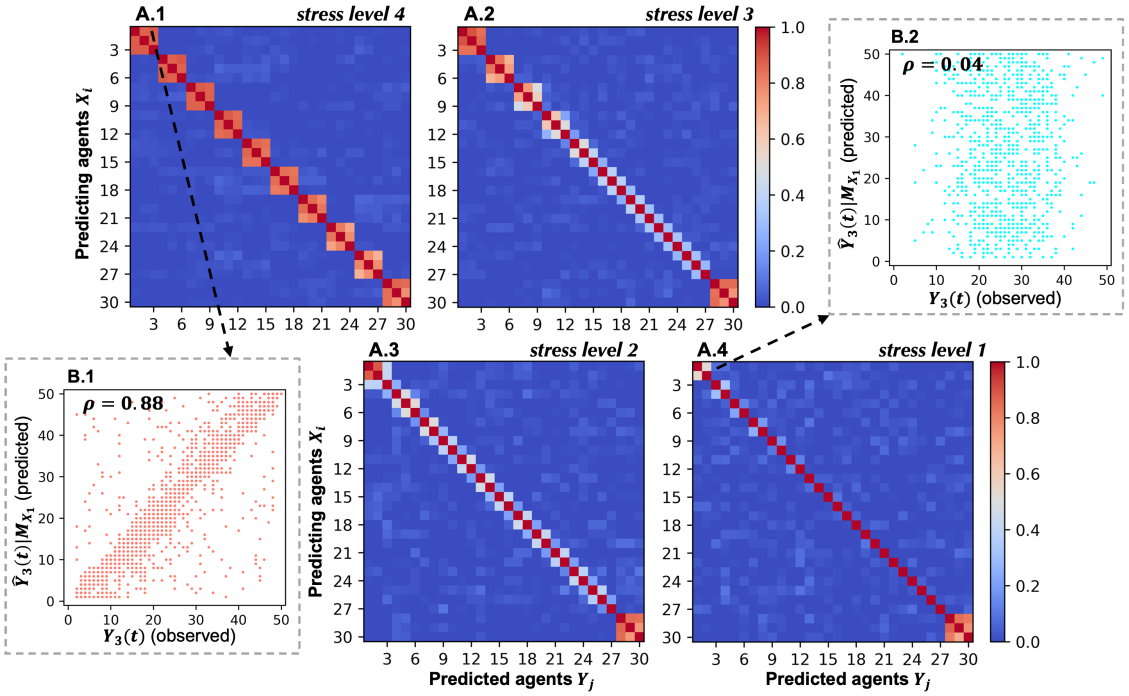


Figure 5.3: (Panels A.1-4) The coordination for 4 stress levels for a team of 30 homogeneous agents in an operation with 50 homogeneous tasks, following cooperation and patrolling strategies. The coordination score $\rho \in [0, 1]$ between two agents is measured by TA-CCM with 1000 episodes. (Panels B.1-2) High and low correlations between observed and predicted decisions of agent 3 given agent 1.

homogeneous tasks (i.e., the dynamics of all task demands are the same) and indexed agents with specialized task-specific capability. The demand level for each task is discrete and within the range of $[0, 4]$. The operation is complete when the demand levels of all tasks reach 0. The capability level of the agent task-specific attribute is 2 for the homogeneous team and can be either $1, 2,$ or 3 for the semi-homogeneous team (with a large number of agents, multiple agents have the same capability level). When two or more agents decide to collaborate on the same task, the joint capability level of the attribute is the sum of all agent capabilities and capped at level 5. A higher capability level reduces the task demand level by 1 with higher probabilities. If the capability level is lower than the task demand level, the task demand level will increase, making the operation extraordinarily hard and requiring synchronized team coordination. It is also assumed that: 1) agents can accurately observe the task level at the assigned task with 80% probability, 2) agents

are able to communicate their local observations with each other, and 3) the communicated observation could be either unaccepted, wrongfully accepted, or accurately accepted for belief update and decision-making.

5.3.1 Decision Strategies

To demonstrate emergent team behaviors and evaluate team coordination, we consider explainable rule-based strategies for teams of homogeneous agents and one RL-trained policy described as the following:

Rule-Based - Agents have accurate full state observations. Lower indexed agents have priority in executing tasks with higher demand levels.

- *Cooperation*: Homogeneous agents form groups of 3 to execute demand level 4 tasks, form groups of 2 to execute tasks with demand level 2 or 3, and individually execute demand level 1 tasks. Agents randomly explore the environment if there are no tasks available.
- *Patrolling*: A group of three homogeneous agents randomly picks three consecutive indexed tasks to execute.

RL-Trained - Agents make decentralized decisions and have partial observations. Agents are trained using Decentralized Deep Q-Learning with Beliefs [27] and make task allocation decisions based on their own past experience and communicated information. Agents are trained to maximize the cumulative discounted reward for completing all tasks with fewer steps. During training, there are no predetermined rules on which agent should execute which task and no restrictions or penalties for switching tasks.

5.3.2 Coordination and Sub-Teams in Rule-based Strategies

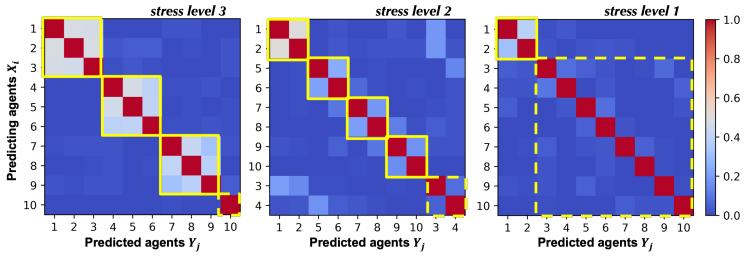
We first investigate the team-coordinated behaviors given predefined rule-based strategies. We consider an operation with 50 homogeneous tasks with a starting demand level at 4 and a team

of 30 homogeneous agents with a capability level of 2. The first 27 agents follow the cooperation strategy and the last three indexed agents follow the patrolling strategy as described in Section 5.3.1. The maximum possible operation demand level the team can experience is 200 (= level 4 $\times$ 50 tasks), and we equally categorize the operation demand levels into 4 ranges $[0, 50], [51, 100], [101, 150], [151, 200]$ that correspond to stress level $1 - 4$ respectively. To collect the task allocation decision time series data, agents play 1000 simulated episodes with up to 50 steps. Data collection stops when all tasks are completed. Decisions at each stress level are aggregated to create a long decision time series.

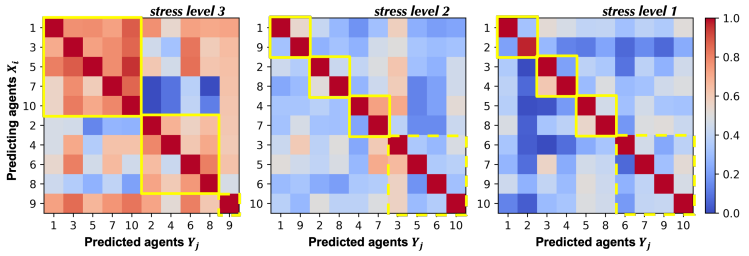
Panels A.1-4 in Fig. 5.3 show the TA-CCM prediction skills between each pair of agents at 4 stress levels. The diagonal entries are always 1 since these are self-predicted scores. By following the defined rule-based strategies, it is known that low-indexed agents choose high-demand-level tasks. When multiple tasks with the same demand level are available, agent groups would choose the tasks randomly. Thus, it is hard to predict agent behaviors across different groups, resulting in low prediction skills in other entries. The last 3 agents always follow the patrolling strategy. Although the patrolling group executes three different consecutive tasks, TA-CCM is able to capture the strong correlations between these three agents. The rest of the agents follow the cooperation strategy. At stress level 4, most of the tasks have demand level 4, and agents almost always form groups of three. At stress levels 3 and 2, tasks have various demand levels, and agents can form groups of any size up to 3. Since lower indexed agents are preferred to execute high-demand level tasks, it is more likely to see high correlation groups (top left corner) in lower indexed agents. At the lowest stress level, most of the tasks are completed, and agents depict random behaviors and low correlations with each other.

Panels B.1-2 in Fig. 5.3 illustrates the prediction skills measured by the correlation ρ between the observed decisions Y_3 and the predicted decisions of agent 3 given agent 1's shadow manifold $\hat{Y_3}|M_{X_1}$ in stress level 4 and 1. Note that TA-CCM is not used for the purpose of predicting the exact decisions of agent 3, it measures the trend and correlation between two agent behaviors. We can conclude that agent 3 behavior can be inferred by using agent 1 behavior more easily, and

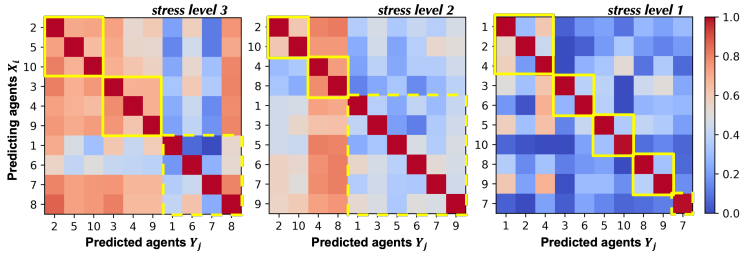
therefore more coordinated at stress level 4 than stress level 1, because agents 1 and 3 are more likely in the same group to execute high demand level tasks and are in different groups to execute low demand level tasks.



(a) Homogeneous agents with Rule-based policy



(b) Homogeneous agents with RL-trained policy



(c) Semi-Homogeneous agents with RL-trained policy

Figure 5.4: Sub-team identification for three 10-agent teams under cooperation rules or RL trained policy in 3 stress levels. Solid yellow boxes indicate identified sub-teams and dashed yellow boxes show ungrouped agents.

5.3.3 Coordination and Sub-Teams in Learned Strategy

In predefined rule-based strategies, we design a priori the agent groups under different task demand conditions. Coordinated behaviors are clearly shown in the prediction skill matrices.

However, with only goal-oriented reinforcement learning agents, it is ambiguous (before training) which agents have statistically significant coordinated decision behaviors and thus can form a sub-team. In this section, we apply TA-CCM and ModMax methods to identify the emergent sub-teams and measure the overall team coordination score for teams of 10 RL-trained decentralized agents in an operation with 14 homogeneous tasks. We consider only 10 agents to reduce the computation time of training, and 14 tasks are challenging enough to create a long decision time series.

We consider 3 different teams and collect their decisions with uncertainties in 2000 episodes. The performance of each team is the average number of steps taken to reduce all task demand levels to 0. The team specifications are summarized in Table 5.1. Teams 1 and 2 both consist of 10 homogeneous agents with capability level 2. Team 3 is semi-homogeneous and has 10 indexed agents with capability levels $[3, 3, 3, 2, 2, 2, 2, 1, 1, 1]$. Since cooperation grants Team 1 full state observation, we add uncertainties in making rational decisions with 20% probability of taking random actions, while RL-trained agents receive accurate local observations with 80% probability.

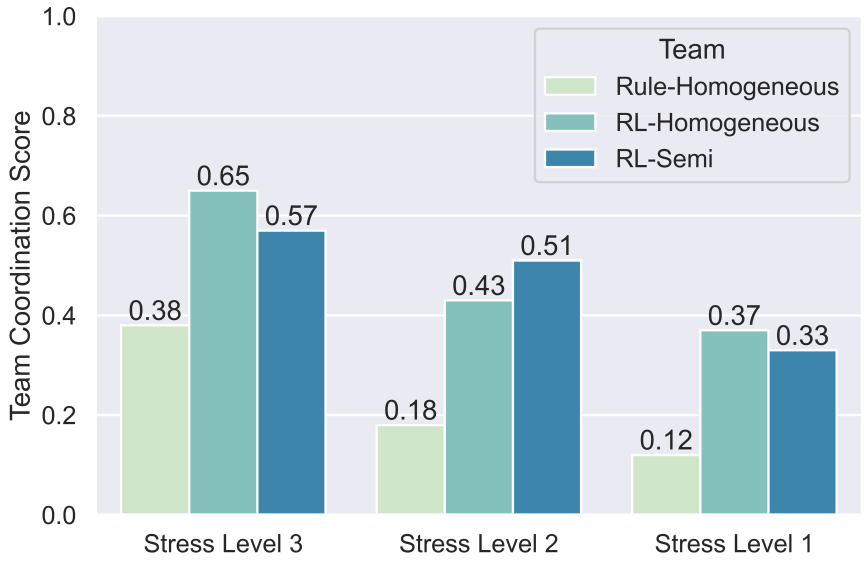


Figure 5.5: Coordination scores for three teams at different stress levels.

Fig. 5.4 shows the prediction skills measured by TA-CCM and sub-teams identified by ModMax for 3 teams. The agent indices are reorganized by the sub-teams. Having 14 tasks with de-

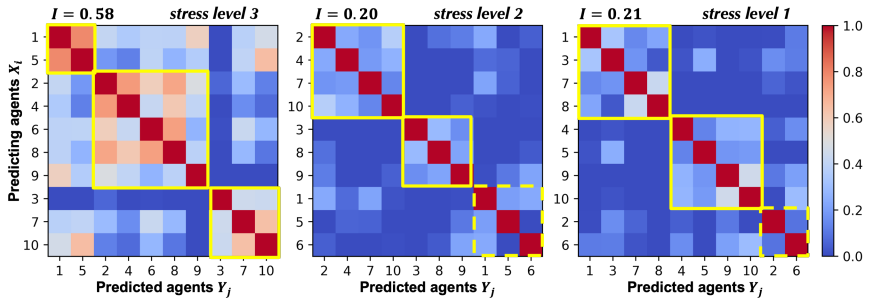
Table 5.1: Team Specifications

Team	Decision Strategy	Uncertainties in Decision/Observation	Steps to Complete All Tasks
1	Rule-based	80% rational	31.6
2	RL-trained	80% accurate	26.4
3	RL-trained	80% accurate	25.7

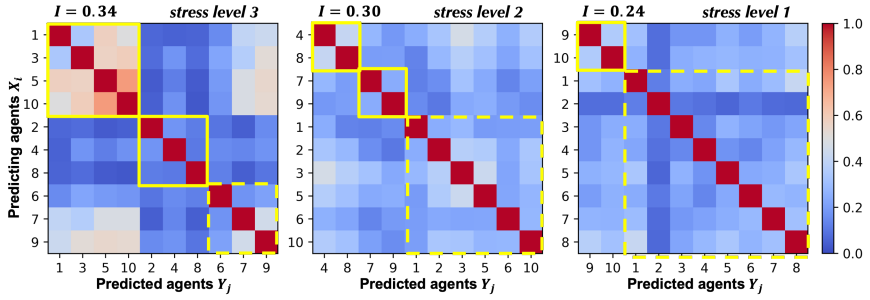
mand level 4, the 3 stress levels correspond to 3 divided task demand level ranges $[0, 18]$, $[19, 37]$, $[38, 56]$. ModMax is able to identify the sub-teams as the predefined groups in cooperation strategy. Since ModMax does not have restrictions on the size and number of sub-teams, it is able to identify larger sub-teams in the RL-trained homogeneous team. Note that we exclude agents that do not belong to any sub-teams, since ungrouped agents have statistically insignificant contributions to team coordination.

Then, we apply the team coordination metric described in 5.2.4 to evaluate the synchronization of three teams. As shown in Fig. 5.5, RL-trained teams illustrate higher coordination than the cooperation team facing an equivalent level of uncertainties, and all teams demonstrate higher coordination in higher stress level situations. Since the cooperation strategy only reacts to the current task situation, it only captures spatial coordination (i.e., agents execute the same task). In contrast, the RL-trained teams maximize the reward for a sequence of actions and are able to capture both temporal (i.e., action sequence) and spatial coordination and consequently are more coordinated. The high coordination score at high-stress levels explains the need for temporal and spatial coordination under severe situations. The coordination scores at lower stress levels imply uncertainty in environment demands and stochasticity of team strategy.

To verify the interpretability of TA-CCM, we illustrate the coordination evaluation in Fig. 5.6, using Pearson correlation and mutual information as coordination measures of each agent pair. Both metrics show symmetric coordination between agent pairs, which neglect the causation effect in agent decisions and therefore reach lower team coordination scores, compared to the RL-Homogeneous Team using TA-CCM. The Pearson correlation can only identify high coordination in stress level 3 because agents tend to work on the same task at high-stress levels, but the estimated



(a) Pearson correlation



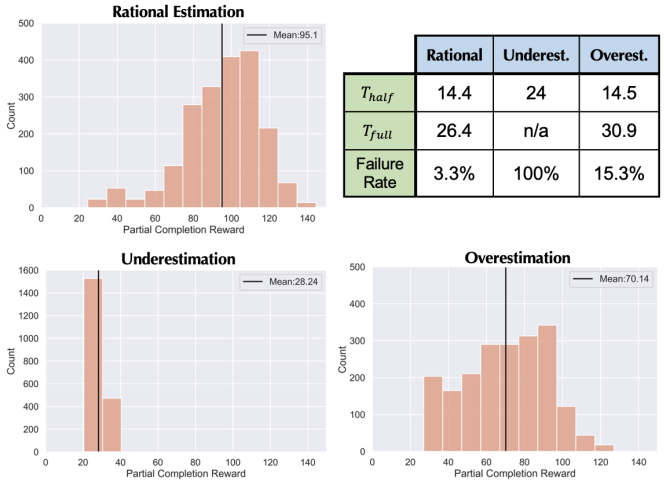
(b) Mutual information

Figure 5.6: Sub-team identification and coordination scores for the 10-agent homogeneous team under RL-trained policy, using (a) Pearson correlation and (b) mutual information.

coordination across different sub-teams and in lower stress levels is much lower than TA-CCM. While mutual information measures how much knowing one variable can reduce the uncertainty of the other, the distribution of an agent's decision does not contain temporal information, and consequently, mutual information fails to identify coordinated sub-teams. Both measures identify different sub-teams and weaker coordination in large sub-teams, compared to TA-CCM.

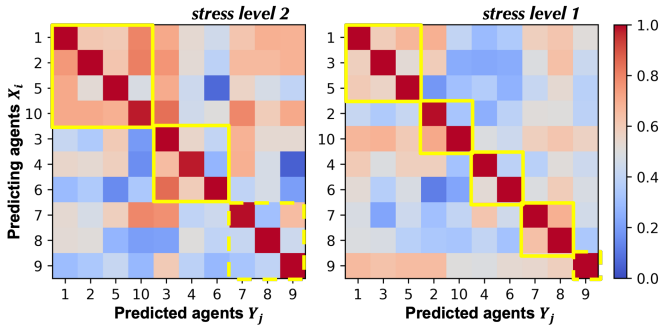
5.3.4 Underestimation and Overestimation

In practice, agents do not always accurately assess the demands and the stress level of the situation, and the dynamics of demands might change due to unexpected events such as degraded agent capability or sudden demand increase. The changes in team behaviors need to be recognized and adjusted accordingly to avoid catastrophic failures. In this study, considering the same operation as in Section 5.3.3, we utilize TA-CCM and ModMax to analyze unexpected coordination strategies when the RL-trained homogeneous team under-/over-estimates the environment task demands. To



	Rational	Underest.	Overest.
T_{half}	14.4	24	14.5
T_{full}	26.4	n/a	30.9
Failure Rate	3.3%	100%	15.3%

(a) Performance averaged over 2000 operation episodes.



(b) Sub-team identification when overestimating. Team coordination score: 0.56 and 0.50 at stress levels 2 and 1 respectively.

Figure 5.7: Performance of the team of 10 RL-trained homogeneous agents when making rational decisions, compared to overestimating and underestimating the stress level.

exaggerate the effect of under-/over-estimation, all agents start to believe that the task demand level is halved/doubled once the environment reaches stress level 2. For example, in underestimation, the agent believes the task demand level is 2 while the actual level is 3 or 4; in overestimation, the agent believes the task demand level is 4 while the actual level is 2. The agent would then make decisions based on the under-/over-estimated demand levels. We consider four performance metrics for under-/over-estimating and making rational decision (accurate estimation) averaged over 2000 episodes as shown in Fig. 5.7a: 1) T_{half}: number of steps to complete half of tasks, 2) T_{full}: number of steps to complete all tasks, 3) Failure Rate: chances of not completing all tasks within 50 steps over 2000 episodes, and 4) Partial Completion Reward: cumulative discounted reward

for completing a portion of tasks (i.e., bringing half of the tasks to level 0 receives half of the completion reward), incentivizing completing tasks in earlier steps. When agents underestimate tasks, they are never able to complete all the tasks. When agents overestimate the situation, they manage to complete all tasks but not as efficiently as making optimal decisions. This is expected since overestimation requires agents to execute the same tasks more often when they could have handled multiple tasks simultaneously. The performance of overestimation is verified through the sub-team identification in Fig. 5.7b. At stress level 1 and 2, the RL-trained homogeneous team demonstrates similar behaviors and coordination scores as at stress level 3 (Fig. 5.4b). Deploying different coordination strategies rather than the trained strategy at different stress levels might result in reduced performance, despite of high coordination score.

5.4 Conclusions

In this chapter, we developed a metric to quantify team coordination in different task demand (stress) levels. The coordination measure was applied to multiple team compositions under various coordination strategies. The approach detects the causality effect between agents in a pair using decision time series data and identifies the highly coordinated sub-teams within the multi-agent teams using a community finding technique in weighted networks. Results of the study show that the proposed measure can identify higher emergent coordination levels for RL-trained strategies across all stages of an operation, and demonstrate that applying a proper coordination pattern at different stages of an operation leads to successful teaming without catastrophic failures.

Understanding the strategies of centralized rules and the evolved strategies of incentive-driven reinforcement learning is fundamental for further improvement in creating large-scale artificial intelligence multi-agent systems, especially in cases where adaptive strategies are needed to ensure the robustness and efficiency of team coordination under unforeseen situations. The coordination metric along with the identified sub-teams can provide the following benefits in evaluating rule-based strategies and training multi-agent RL agents: 1) detecting whether the learned strategy and sub-team formation are proper in corresponding situations, 2) monitoring whether the team and the

sub-teams can maintain high-level coordination when situations become unpredictable, 3) potentially simplifying team design with sub-teams, and 4) improving the coordinated learning by using coordination score as guidance. The proposed method provides opportunities to acknowledge how communities (highly coordinated sub-teams) or leader-follower (one-directional causality) relations are forming and influencing coordination strategies in various types of multi-agent teams.

5.5 Algorithmic Recourse — Application of Reinforcement Learning in Explainable AI

Explainable AI (XAI) refers to approaches that enable humans to comprehend and trust the results and outputs created by AI/machine learning (ML) algorithms. Unlike traditional "black box" models (e.g., artificial neural networks, transformers), which offer little insight into their logical reasoning processes, XAI aims to provide clear, interpretable, and transparent explanations of how AI systems make decisions or how ML algorithms generate certain outputs. This approach not only enhances trust and accountability but also assists in debugging and improving model performance. This chapter develops an interpretation technique to explain the coordination strategies that are the output of Q-networks trained through reinforcement learning. Such a behavior explanation is considered post-hoc, as it does not influence the training process. In Appendix A, we present another post-hoc explanation technique, algorithmic recourse, that is model-agnostic (i.e., a generalized method to explain any type of AI/ML systems) and applicable in any problem domain.

AI/ML continues to be integrated into areas that might lead to life-critical events such as healthcare diagnosis, insurance pricing, and financial loans, and many business and operation problems (e.g., marketing, product launching, fundraising, etc.) pivot to relying on evidence-based and data-driven insights. The need for explainable models becomes increasingly crucial to ensure ethical standards, regulatory compliance, and equitable services. When an individual is affected adversely by the decisions of AI/ML systems (i.e., unsatisfactory outcomes), explanations on *why* certain decisions are made and recommendations on *how* to reverse the outcome are needed. Algorithmic

recourse [143, 144] refers to a computed recommendation provided to an end user that suggests specific feature changes they can make to convert an unsatisfactory outcome into a favorable one. Sequential algorithmic recourse methods [145, 146] recommend a series of changes (or a path) that lead to a favorable outcome with minimal cost. These methods focus on action feasibility and the proximity of feature changes to determine the cost of actions. However, the uncertainties of feature changes and the risk of higher-than-average costs in recourse have not been considered. It is undesirable if a recourse could (with some probability) result in a worse situation from which the recovery requires an extremely high cost, and sometimes such a recourse may even be dangerous to suggest. For example, asking a person to change jobs may result in losing their current job and requiring more effort to recover. Therefore, it is essential to incorporate risks when computing and evaluating recourse. Finding options associated with lower risks but a slightly higher cost may be preferred by an individual, and a detoured / long-distance path of actions may be a better recourse option if the actions required carry less risk. We call the recourse computed with such risk considerations as Safe Algorithmic Recourse (SafeAR). The objective is to empower people to choose a recourse based on their risk tolerance.

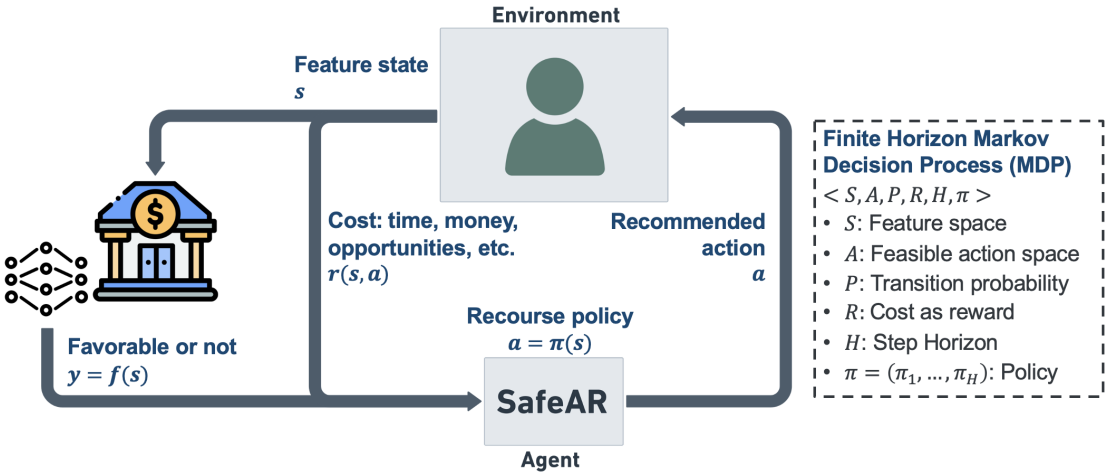


Figure 5.8: Reinforcement learning formulation for algorithmic recourse.

Moreover, we discuss and show how existing recourse desiderata can fail to capture the risk of higher costs, and we present a method to compute recourse policies that consider variability in

cost by connecting algorithmic recourse literature with risk-sensitive reinforcement learning. The objectives of SafeAR are to suggest different recourse policies with different risk profiles and to empower the affected individuals with risk-averse options to choose from. As shown in Figure 5.8, the recourse problem can be formulated into a finite horizon Markov Decision Process and solved by Reinforcement learning (RL), defined as State S, Action A, Transition P, Reward R, and Step Horizon H. The resultant policy is referred to as a recourse policy π. Given the current feature state of an individual s, the SafeAR agent suggests an action a for the individual to take. By taking this action, there is an associated cost r, and the feature state would change to another state s' based on the success likelihood as transition probability p. Then the SafeAR agent would suggest the next action following the recourse policy π. π finds the best action to take given any feature state, which maximizes the expected reward (or minimizes the cost) and can incorporate uncertainty in cost and action effects. To account for risk, we incorporate the variance in costs during policy computation and connect risk-sensitive reinforcement learning ideas [147–149] with algorithmic recourse. In recourse, there is another element. A machine learning model that determines whether the feature state gives a favorable outcome. In the RL formulation, these feature states are goal states. They signal the success of the recourse, and the cost would be zero at the goal states, as no more action is needed. We also adopt measures "Value at Risk" [150] and "Conditional Value at Risk" [151] from the financial literature to summarize risk concisely and visualize risks during recourse. We apply our method to real-world datasets related to health insurance premiums, income, and financial loans, and compare policies with different risk-aversion levels. We show that risk-aware recourse policies can provide better risk guarantees, compared to risk-neural recourse recommendations. In addition, we initiate an investigation into the disparities in terms of risk exposure during recourse and observe the existence of discrepancies between gender groups in risk measures for the same risk-aversion level, which motivates further studies on recourse fairness.

CHAPTER 6

Multiagent Inverse Reinforcement Learning via Theory of Mind Reasoning

The behavior interpretation technique discussed in Chapter 5 uncovers the coordination pattern among autonomous teammates. This chapter, as a step further, utilizes the insights gained from behavior interpretation to develop assistive AI for humans in collaborative operations that provide decision recommendations with social cognition capabilities.

6.1 Introduction

Understanding the underlying strategies of how humans interact with each other in collaborative teaming scenarios is a vital component of creating systems that appropriately predict team behavior, understand collaboration dynamics, and provide assistance to potentially improve performance. This is particularly challenging when people are collaborating for the first time and therefore know little about the goals, strategies, and intentions of others, yet have to adapt to each other as they perform a cooperative task.

One way to predict what individuals will do is to create models of their decision-making process given traces of their behavior, using machine learning techniques. One approach is to learn a policy, i.e., a mapping from states to actions, directly from observation. This includes imitation learning techniques, e.g., behavior cloning [152] that has the power of replicating the behavior of others without the need for environment and dynamics modeling [153]. However, such approaches lack the capability of understanding the underlying intentions of individuals or the goals of the task.

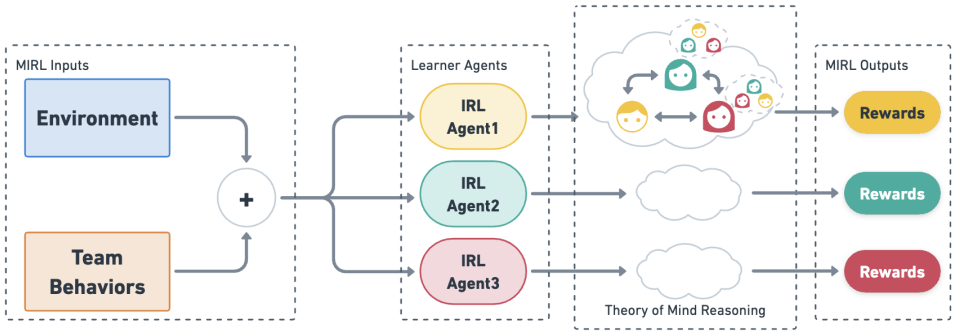


Figure 6.1: Overview of the proposed approach for Multiagent Inverse Reinforcement Learning via Theory of Mind Reasoning (MIRL-ToM). Given the environment and the demonstrated behaviors of a team of humans, each learner agent performs single-agent IRL by reasoning about other agents' behavior via ToM and output the resulting reward functions.

Further, in teamwork settings, knowing the policy of individuals alone makes it hard to understand how they coordinate their behavior [154], or whether there are impediments to cooperation, e.g., a mismatch between the preferences of each team member.

An alternative, more robust method is to assume that the behavior of each individual is the result of a Reinforcement Learning (RL) process, where agents learn how to perform a task through trial-and-error interactions with a dynamic and uncertain environment [155]. The goal of an RL agent is specified by a reward function that encodes its preferences and whose output the agent wants to maximize over its lifetime. Therefore, understanding the underlying strategy of an individual by observing its behavior relies on recovering the reward function guiding the behavior during task performance. This problem is known as Inverse Reinforcement Learning (IRL) [52, 53] and involves inferring the reward function used by an individual, referred to as the *expert*, given traces of its behavior. IRL has advantages compared to imitation learning approaches in that the reward function is the most succinct and robust representation of the expert's goals and allows for the prediction of behavior in novel situations or different tasks (generalization) [52].

Within IRL, apprenticeship learning [156] established a linear programming method to match the learned reward model to the feature counts derived from state-action pairs in the expert's demonstrations. A game-theoretic approach [157] was also proposed to learn from playing repeated games even in the absence of experts. Maximum margin IRL [158] formulates a quadratic

problem to find the reward solution producing behavior paths that are closer to the expert's behavior than others by a margin that scales with the path difference loss. One of the main challenges of IRL that is addressed by these approaches is that the IRL problem is ill-posed — the same observed behavior can be explained by multiple reward functions, and the same reward function can lead to different behaviors under stochastic dynamics [53]. To resolve this ambiguity, many IRL algorithms provide probabilistic formulations by finding the reward function that maximizes the likelihood of the demonstrated trajectories. In particular, Maximum entropy (MaxEnt) IRL [97] finds a trajectory distribution where the expert demonstrations are exponentially preferred. Bayesian IRL [133] presents an efficient sampling method to infer the posterior distributions of reward function parameters.

Understanding the strategies and intentions underlying the behavior of a team in collaborative tasks requires extending IRL to multiagent settings, which is known as Multiagent Inverse Reinforcement Learning (MIRL). In MIRL, the goal is to infer the individual reward functions guiding the behavior of each agent, given trajectories of a team's (joint) behavior. It is often assumed that the joint behavior is the result of an equilibrium solution concept, such as Nash equilibrium [54,55]. This makes the MIRL problem harder because, in addition to multiple reasonable reward functions given the demonstrations, there may be multiple equilibria that are a solution to a combination of individual rewards [55, 56].

Many of the existing approaches to MIRL assume the existence of a centralized learner that computes the joint action equilibrium given the (known) reward functions of each individual [56, 159, 160]. This implies that all individuals perfectly know each other's preferences (task payoffs/reward functions) at all times and then play according to the equilibrium solution. However, assuming perfect information about others in a team might be too restrictive. First, the individuals might be interacting with unknown teammates, e.g., the observed behavior was produced by individuals recruited to perform a collaborative task who tried to adapt to each other's behaviors while performing the task. Second, the assumptions formed by each individual regarding the teammates' goals and intentions, created by observing their behavior, might be incorrect due to the noisy nature

of human behavior. Finally, because individuals are adapting their behavior to that of their team members (non-stationarity), the assumptions about their preferences must constantly be updated as a result of the individuals' interaction. All of this makes it less likely that a team's demonstrated behavior was derived from a unique, stationary, known a priori equilibrium.

To address the aforementioned issues, we borrow the concept Theory of Mind (ToM) from cognitive science [59] which ascribes mental states to explain and predict the actions of others. In particular, we model each individual as adapting to the behavior of the teammates by maintaining a probability distribution over *baseline behavior profiles* for each other member of the team, which is updated as the task unfolds and interactions between team members occur. Computationally, these profiles correspond to domain-specific reward models encoding typified preferences in a task. We then split MIRL into two phases (see overview in Fig. 6.1):

1. *Model inference:* for each individual in the team, we use ToM reasoning to estimate a posterior distribution over baseline profiles for each other teammate given their behavior included in the trajectories — this models how each team member might have modeled each other by observing their behavior during task performance.

2. *Decentralized MIRL:* We perform MIRL via a decentralized equilibrium approach where we break down the MIRL problem into multiple single-agent IRL problems similar to the approach in [55]. A reward function is inferred given the corresponding individual's demonstrated behavior via MaxEnt IRL, where we simulate the behavior of other team members according to the time-varying distribution over baseline profiles computed in Phase 1 — this models best-response strategies conditioned on the perceived models of others.

The analogy between IRL and ToM has been drawn in [161] to link the mental states of humans to the reward models of IRL agents [162]. Despite applications of ToM in multiagent RL [163], limited research has been done to investigate the underlying strategic behaviors and motivations of teams in the MIRL setting. The main contribution of this chapter is a decentralized MIRL framework using ToM reasoning (MIRL-ToM) that extends MaxEnt IRL to multiagent settings.

Our framework allows modeling interactions between arbitrarily unknown teammates under the assumption of a set of baseline reward profiles. We deal with uncertainties in the intentions of other teammates by performing ToM reasoning over the profiles in a Bayesian setting.

We evaluate our approach in a simulated 2-player search-and-rescue task where each agent plays a different role with a distinct set of skills in the task. We designed rewards for each agent that include both task- and social-related aspects allowing for different types of coordination strategies to emerge. We assess the capability of our MIRL-ToM approach in recovering the original reward functions of agents given different sets of baseline profiles. Overall, our results show that the choice of baseline profiles in Phase 1 is paramount to the recovery of ground-truth rewards in phase 2. In particular, including baseline profiles corresponding to the opposite goals of team members prevents the convergence of MaxEnt IRL, while including baselines whose goals are close to those of the original teammates leads to reward functions similar to the ground truth. Further, we show that our method, using ToM reasoning over the demonstrated behaviors, can be used to recover the rewards used by agents interacting both with known and unknown teammates, which attests to the robustness of our approach.

Generalizing IRL to multiagent systems is hard because we aim to learn individual reward functions that explain both how each agent behaves—which involves recovering the rewards via IRL—and how together the agents coordinate— which involves modeling the joint behavior as a solution to an equilibrium concept. Within MIRL, in [164] an algorithm was developed to train a centralized agent that optimizes coordination of behaviors for traffic-routing problems; however, while learning the rewards via IRL, the agents ignore their teammates which are treated as part of the environment. In [56], Bayesian inference is performed on zero-sum two-person games by optimizing rewards under a likelihood function that encodes the notion of a minimax equilibrium, but the solution is not scalable to n-player games and complex domains. Further, many MIRL solutions rely on computing some notion of equilibrium given the individual reward functions at each iteration. Some approaches deal with competitive games by learning Markov perfect equilibrium policies given individual rewards [159], playing against Nash Equilibrium policies through

adversarial training [160], evaluating undesirable strategic behavior by extracting utility functions from demonstrations [55], and leveraging Generative Adversarial Networks [58] to retrieve reward functions in high-dimensional domains [57]. However, these methods still require finding an equilibrium policy throughout learning, and it is assumed that experts have perfect knowledge of the goals and intentions of their teammates, ignoring the dynamic nature of interactions and the possibility of first encounters. The proposed MIRL-ToM, combining ToM reasoning with single-agent MaxEnt IRL, is able to not only decentralize the model reasoning and reward learning process but also consider situations of unknown teammates and adaptation via belief updates over a set of baseline reward profiles.

6.1.1 Reinforcement Learning

Reinforcement learning (RL) [155] relies on the formalism of (single-agent) Markov Decision Process (MDP), which is defined as a tuple of $\langle S, A, T, R, \gamma \rangle$, where $s \in S$ is a finite set of states, $a \in A$ is a finite set of actions, $T := Pr(s', a, s)$ is the transition probability between states given an action, $R := R(s, a)$ is the reward of reaching a state and performing an action, and $\gamma \in [0, 1)$ is the discount factor that encourages receiving larger reward at earlier steps. Under the Bellman equations, the optimal quality value of an action given a state satisfies $Q^*(s, a) = R(s, a) + \gamma \mathbb{E}_{s' \sim T}[maxQ^*(s', a')]$. (Forward) RL algorithms find the optimal policy $\pi : S \rightarrow A$ that can be retrieved using $\pi^* = argmax_a Q^*(s, a)$.

6.1.2 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) [52] represents the inverse problem of RL, i.e., finding the reward function R of the learner agent given S, A, T and a collection of demonstrated trajectories $\mathcal{D} = \{\zeta_0 \dots \zeta_i \dots\}$ following the internal policy of an expert. Each trajectory $\zeta_i = \{(s_0, a_0) \dots (s_j, a_j) \dots (s_t, a_t)\}$ consists of state-action pairs over t steps. IRL aims to find the reward function parameters that maximize the likelihood of demonstrations. Maximum Entropy IRL [97] (MaxEnt IRL) utilizes the principle of maximum entropy and assumes that the demon-

strated action given a state is exponentially preferred over other actions. MaxEnt IRL considers the trajectory reward $R(\zeta)$ as a linear combination of state feature counts (i.e., $R(\zeta) = \theta^T \phi(\zeta)$) and iteratively finds the parameters $\theta^* = \arg\max_\theta \mathbb{E}_{\zeta \sim Pr(\zeta|\theta)}[\sum_j r(s_j, a_j|\theta)]$ that best reproduce the demonstrated behaviors such that the probability of taking a trajectory ζ is proportional to the exponential of the rewards along the path (i.e., $Pr(\zeta|\theta) \propto e^{R(\zeta)}$).

6.1.3 Multiagent Inverse Reinforcement Learning

Multiagent Inverse Reinforcement Learning (MIRL) extends IRL to multiagent settings. The collected team trajectories are denoted as $\mathcal{D} = \{\zeta_0 \dots \zeta_i \dots\}$, where $\zeta_i = \{(s_0, \hat{a}_0) \dots (s_j, \hat{a}_j) \dots (s_t, \hat{a}_t)\}$ is each team trajectory in the form of state-joint action pairs $(s_j, \hat{a}_j)$ over t steps, where $\hat{a}_j^k$ denotes the action of agent k at step j. MIRL then aims to recover a set of reward function parameters for all agents $\Theta^* = \{\theta^*\} = \arg\max_\Theta \mathbb{E}_{\zeta \sim Pr(\zeta|\Theta)}[\sum_j r(s_j, \hat{a}_j|\Theta)]$ that best match the collected team trajectories.

6.1.4 PsychSim

We implemented our approach using PsychSim [165], a framework that models decision-theoretic agents with recursive Theory-of-Mind (ToM) for interactive social simulation [165]. To achieve this, PsychSim models world dynamics under the Multiagent Partially-Observable Markov Decision Process (MPOMDP) formalism [166, 167]. At each step, each agent selects an action based on the values computed for each available action by assessing risk/reward trade-offs in decisions up to a specified planning horizon (cumulative discounted reward), conditioned on the actions of others. Namely, during planning each agent computes the action values for all other agents using recursive ToM, where others are considered reward maximizers under the (mental) models that the agent has of them. To model uncertainty, agents can form beliefs about the state of the environment and the models of other agents, which are updated via Bayesian inference based on observations of state features and the actions of others, respectively.

6.2 Methods

We assume that individuals in a team coordinate their behavior without explicit communication. In such cases, they need to reason about the goals and intentions of others while computing their own actions. We also assume the existence of a set of *baseline agent profiles* that encode domain-specific, notional preferences in the task of interest. For example, in a search-and-rescue task, this might correspond to having profiles for trying to search for victims, triage them, calling for backup, etc. Before a task, an individual might assume that teammates behave according to a prior distribution over the baseline profiles, e.g., uniform, or a preference given to a profile given background information. As the task progresses and individuals observe the behavior of others, this distribution is updated accordingly. We propose Multiagent Inverse Reinforcement Learning via Theory of Mind (MIRL-ToM) to allow reasoning about others' behavior while performing a task, corresponding to a decentralized approach to learn the reward functions of each team member. The approach consists of two phases as shown in Fig. 6.2, the details of which are discussed in the following sections.

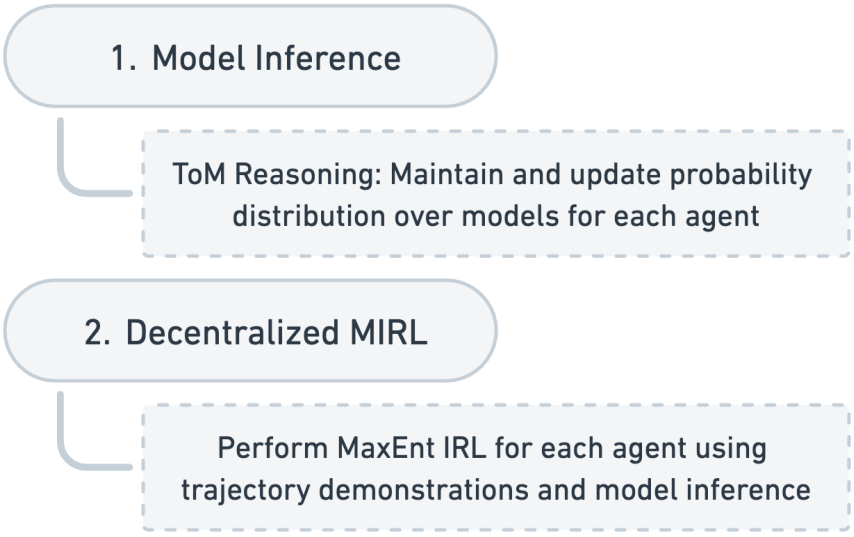


Figure 6.2: Two phases of the Multiagent Inverse Reinforcement Learning via Theory of Mind.

6.2.1 Model Inference

In the model inference phase, we use ToM reasoning to update a probability distribution (belief) over baseline profiles given the behavior of an agent in the trajectories. Before the inference process, we pre-select the set of baseline profiles for each agent. In general, the profiles can encode any agent decision-making *model*, and we denote the initial profile set by $\{m_i\}$. In this chapter, we consider profiles to correspond to predefined reward functions, i.e., reward profiles. Let us denote by $b^k(m)$ the prior distribution that an agent holds over the models of another agent k. For each step of each trajectory ζ_i, $b^k(m)$ is updated via Bayesian inference:

$$b'(m) \propto Pr(a|m, \zeta) \times b(m|\zeta) \tag{6.1}$$

where we drop the agent indexing k for convenience. This phase augments the team trajectories, $\hat{\mathcal{D}} = \{\hat{\zeta}_i\}$, by adding the agent's belief over agent models at each step, resulting in a set of state-joint action-model distribution tuple sequences: $\hat{\zeta}_i = \{(s_j, \hat{a}_j, b_j(m)\}$. In loose terms, this phase captures how the intentions of agents could be modeled by others given their actions in the trajectories, conditioned on a set of agent profiles and the prior distribution.[1]

6.2.2 Decentralized MIRL

After each learner agent computes the (time-varying) beliefs about the expected reward models of its teammates, we perform decentralized MIRL by computing a reward function for each individual agent separately. Without loss of generality, here we assume that the reward functions $R(s, a) := \theta^T \phi(s, \hat{a})$ are linear combinations of (potentially nonlinear) reward features $\phi(s, \hat{a})$ parameterized by reward weights θ. We follow the maximum entropy principle to find the individual reward function where the demonstrated trajectories of the agent are exponentially preferred [97],

[1] We note that the initial set of profiles and the prior distribution need not to be the same for all observing agents. Similarly, the belief update process can be different for distinct agents observing the same teammate, e.g., by restricting observability to local interactions between agents.

i.e.:

$$\theta^* = \arg\max_{\theta} L\left(\theta | b(m)\right) = \arg\max_{\theta} \sum_{\zeta_i \in \mathcal{D}} \log Pr\left(\zeta_i | \theta, b(m)\right) \tag{6.2}$$

Algorithm 3 Decentralized MIRL via ToM

1: $\mathcal{D} \leftarrow$ Collection of team trajectories
2: **for** each learner agent k **do**
3: $b_0(m)$: prior model distribution, $\phi(s,a)$: feature vector
4: $\hat{\mathcal{D}} = \{\hat{\zeta}_i\} \leftarrow$ Model inference on $\mathcal{D}$ using ToM reasoning
5: $\phi^k_{emp} \leftarrow \sum_{\zeta_i \in \mathcal{D}} \sum_{(s_j, \hat{a}^k_j) \in \zeta_i} \phi(s_j, \hat{a}^k_j)$
6: Initialize reward vector weights θ^k
7: **while** θ^k not converging **do**
8: $\pi^k(a|s, b(m), \theta^k) \leftarrow$ forward pass
9: $\phi^k_{est} \leftarrow \sum_{\zeta_i \sim \pi} \sum_{(s_j, \hat{a}^k_j | b_j(m)) \in \zeta_i} Pr(s_j | \pi^k, \theta^k) \phi(s_j, \hat{a}^k_j)$
10: $\nabla L(\theta^k) \leftarrow \phi^k_{est} - \phi^k_{emp}$
11: $\theta^k \leftarrow \theta^k - \alpha \nabla L(\theta^k)$
12: **end while**
13: **end for**
14: **return** $\{\theta^k\}$

We propose an algorithm that extends the MaxEnt IRL [97] to the MIRL setting via decentralized equilibrium computation, where at each iteration, we approximate the distribution over paths of an agent conditioned on both the current reward weight and the belief over agent models computed in the model inference phase, as shown in Alg. 3. For ease of explanation, in Alg. 3 we show the case where the team contains only two agents, i.e., there is only one model distribution, $b(m)$, to be computed. To solve the MIRL problem, we learn a reward weight vector θ^k for each individual agent k by iteratively improving according to whether the weight vector leads to the behavior similar to that exhibited by the corresponding expert in the demonstrated trajectories, as measured by comparing the estimated feature counts (FCs) against the empirical FCs (Alg. 3, line 5). Therefore, we achieve a decentralized equilibrium by having each individual reward function be the solution to the best response strategy to the perceived models of the other agents during estimated FCs computation. We note that, as an advantage to this approach, the learning process for each agent can be completely distributed and parallelized. At each iteration, we assume the existence of a forward RL solver computing the policy π^k given the current reward weight vector

candidate, θ^k, where the actions of the other agent are optimized according to the distribution over models $b(m)$ at each timestep (Alg. 3, line 8). During the gradient descent step (Alg. 3, lines 9-11), the estimated FCs ϕ_{est} are computed by sampling Monte Carlo trajectories using the resulting joint policy. This process can be visualized in Fig. 6.3.

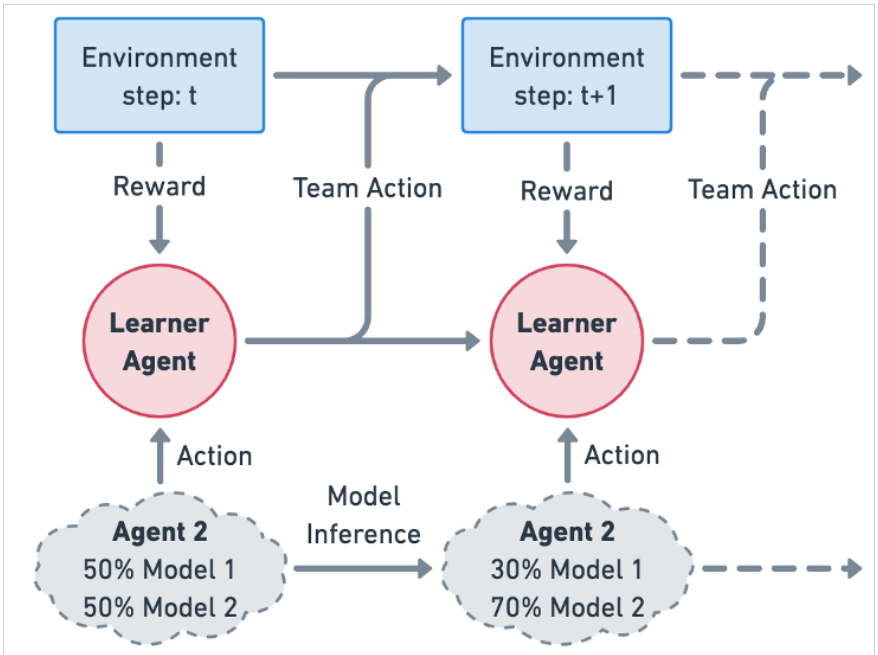


Figure 6.3: Forward pass of feature count estimation. Each learner agent uses the model distribution data to select the actions of other agents and select an action given the current reward model. The feature counts are computed from the environment states and the team actions.

6.3 Results

To assess the effectiveness of the proposed MIRL-ToM approach, we consider a collaborative environment and a team of two agents. Instead of evaluating our approach with real human data, where the underlying reward functions cannot be accessed, here we specify a set of ground-truth rewards, collect trajectories by having the agents maximize those rewards according to some predefined model of others, and then compare the learned to the ground-truth reward functions for each agent. This is a necessary first step to assess the capabilities (and limitations) of our approach

before testing with human data.[2]

6.3.1 Experiment

6.3.1.1 Environment

We designed a collaborative multiagent search-and-rescue (S&R) operation using PsychSim [165] which allows the modeling of partial observations, stochastic state-action dynamics, and ToM reasoning via agent modeling. In our S&R task, two agents need to cooperate in order to search all locations in a gridworld environment and evacuate all victims as quickly as possible. Initially, the victims' locations are unknown to the agents, which we model via a uniform belief over the existence of victims in each location, which is set to the real value once agents enter the corresponding cell. The state of the environment $s \in S := S_1^P \times \cdots \times S_n^P$ describes a grid world with n locations, where each location has a feature $v \in V := \{unknown, found, ready, clear, empty\}$ denoting the victim status at that location. The action set includes three types of actions: 1) movement actions: $\{up, down, left, right, wait\}$ that deterministically move the agent in the corresponding direction, 2) victim-handling actions: $\{search, triage, evacuate\}$ that changes the victim status feature, and 3) communication action, $call$, which allows an agent to incentivize others to come to its location. The state transition of the victims' status is deterministic, the dynamics of which are illustrated in Fig. 6.4. As denoted by the red arrow, to model the need for cooperation, we require the two agents to be at the same location and perform action $evacuate$ at the same time for a victim to be evacuated.

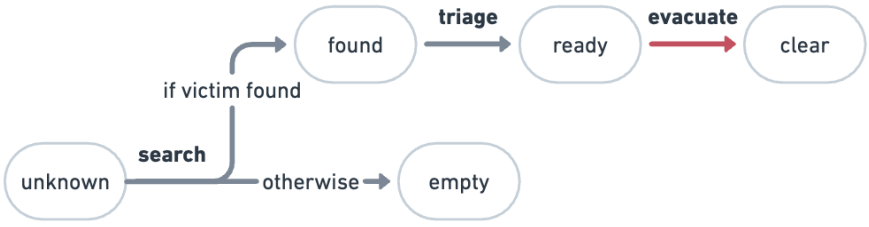


Figure 6.4: Transition dynamics of the victim status. $\boldsymbol{search}$ either finds a victim or an empty cell, $\boldsymbol{triage}$ prepares a victim, and it requires two agents at the same location to $\boldsymbol{evacuate}$ it.

[2]Our implementation and experiments are available at https://github.com/usc-psychsim/mirl-tom-aamas23.

Table 6.1: Ground-truth reward functions for the two agents as linear combinations of features ϕ_i weighted by θ_i.

Role-Related Tasks Social/Helping

$\phi_i(s,a)$	θ_i	**Description**
Dist2Vic	0.06	Distance to the nearest $found$ victim
Search	0.06	Whether to search $unknown$ locations
Triage	0.19	Whether to triage $found$ victims
Evacuate	0.63	Whether to evacuate $ready$ victims
Wait	0.03	Whether to standby
Call	0.03	Whether to call for help if needed

(a) Medic

$\phi_i(s,a)$	θ_i	**Description**
Dist2Help	0.25	Distance to the agent with $call$ action
Search	0.25	Whether to search $unknown$ locations
Evacuate	0.50	Whether to evacuate $ready$ victims

(b) Explorer

6.3.1.2 Ground-Truth Agents (Experts)

We consider a team of two agents with designated roles—Medic and Explorer—each defining different action sets and reward functions. The Medic agent has the full action set, while the Explorer agent is not able to $wait$, $triage$, or $call$, i.e., it can only *explore* the environment. The reward function of each agent $R(s,a) := \sum_i \theta_i \phi_i(s,a)$ is linearly parameterized by ground truth weights θ as defined in Table 6.1. The distance features compute the Manhattan distance from the agent to the target location. The action-related features are indicator functions that denote whether an action was taken in the current state. As seen in Table 6.1, reward features are categorized into role-related *task* features (orange) and *social* features (blue). We model the Medic as prioritizing the rescue of victims, while the Explorer prioritizes searching but also reacts to the call for help from the Medic.

6.3.2 Collecting and Visualizing Trajectories

Given the ground-truth reward functions defined in Table 6.1, we start with the case where the two agents interact with each other and perform the task knowing exactly each other's reward function (perfect models). Notwithstanding, we note that the MIRL-ToM procedure is unaware of this and is only given the generated team trajectories and a set of baseline profiles. The goal is to test the importance of different priors over profiles for the recovery of the original rewards. We collected 16 trajectories, each with 25 steps, for the team of agents, produced by having the agents plan with a horizon of 2 using a soft-max action selection. Fig. 6.5 shows the collaborative behavior of two agents at three different steps in one example trajectory. At $t = 4$, the Medic calls for help, and the Explorer moves toward the Medic instead of searching or moving away; $t = 10$ illustrates role-dependent actions: the Medic is waiting, because all victims are either unknown or clear, while the Explorer keeps searching other locations; at $t = 16$, both agents evacuate the victim together.[3]

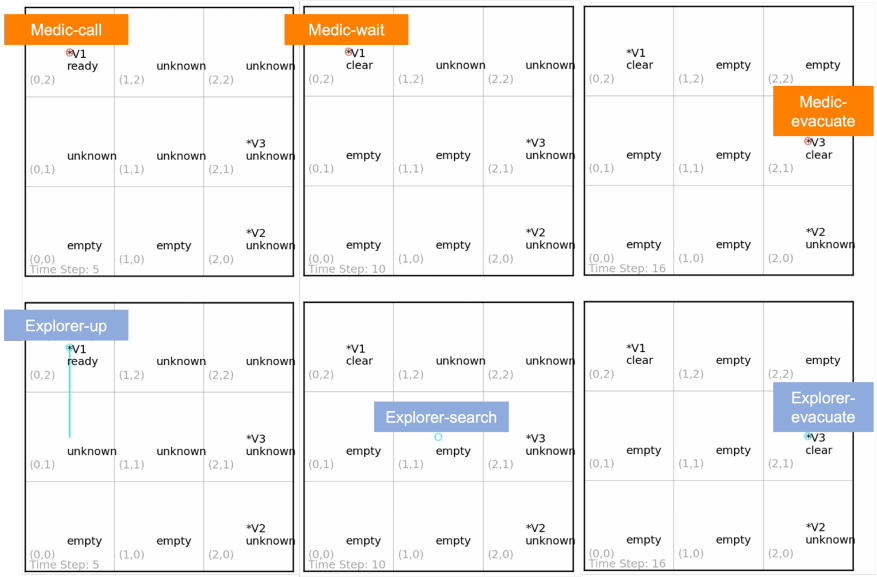


Figure 6.5: An example trajectory at timestep $\boldsymbol{t = 4}$ (left), $\boldsymbol{t = 10}$ (center), and $\boldsymbol{t = 16}$ (right) for the Medic (top) and Explorer (bottom) agents.

[3]Example simulated animations showing the agents interacting in the S&R environment are included HERE as the supplementary material.

6.3.3 Model Inference Results

Based on the reward features ϕ_i detailed in Sec. 6.3.1.2, we designed a set of baseline profiles as shown in Table 6.2 by creating different variations of the ground-truth rewards and the task- and social-oriented features. The goal was to create a set of reward functions one would typically expect individuals to follow in our S&R task.

From these reward profiles, we define the following experimental conditions, i.e., each defining the set of agent models/profiles used in the Model Inference phase (see Sec. 6.3.3):

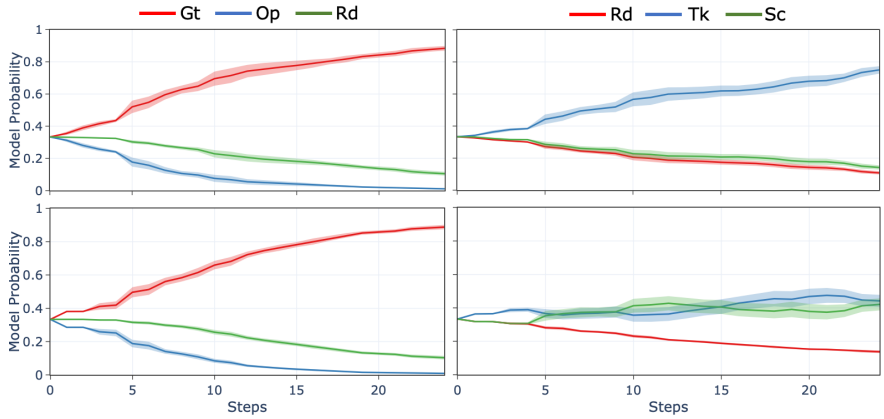


Figure 6.6: Average model probabilities with standard deviations (shaded) of 16 trajectories for Medic (top) and Explorer (bottom) agents under two experimental conditions: ***cond***-3 (Gt-Op-Rd, left) and ***cond***-4 (Rd-Tk-Sc, right).

Table 6.2: Baseline reward profiles

	Reward Function	Description
Gt	θ_{gt}	Ground truth rewards
Op	$-\theta_{gt}$	Opposite goals
Rd	θ_0	No goals / random behavior
Tk	$[\theta_{task}, \theta_{social} = 0]$	Focus on task-specific aspects
Sc	$[\theta_{task} = 0, \theta_{social}]$	Focus on social aspects

***cond*-1: Gt,** always believe the other agent uses Gt rewards.

***cond*-2: Op,** always believe the other agent uses Op rewards.

***cond*-3: Gt-Op-Rd,** update belief $b(\{Gt, Op, Rd\})$ via ToM.

***cond*-4: Rd-Tk-Sc,** update belief $b(\{Rd, Tk, Sc\})$ via ToM.

Condition $cond$-1 corresponds to the setting where the learner agent uses the ground-truth rewards of the other (perfect model) to compute the estimated FCs during MaxEnt IRL. The rationale is to assess the best-response (decentralized equilibrium) computation during single-agent IRL. $cond$-2 presents a situation where the learner agent always believes its teammate has goals opposite to those of the ground-truth agent. Here we assess the importance of agent co-dependencies in the task and test the limits of MIRL in recovering the original rewards given contradictory models of others. For these first two conditions, the model distribution is fixed so we do not perform model inference via ToM reasoning. In contrast, $cond$-3 and $cond$-4 both use ToM reasoning. $cond$-3 verifies the accuracy of the model inference procedure in identifying the ground truth model given the observed behavior, while $cond$-4 investigates whether MIRL-ToM can still recover rewards close to the original without being given the Gt model given, i.e., such as it might happen when analyzing real human data.

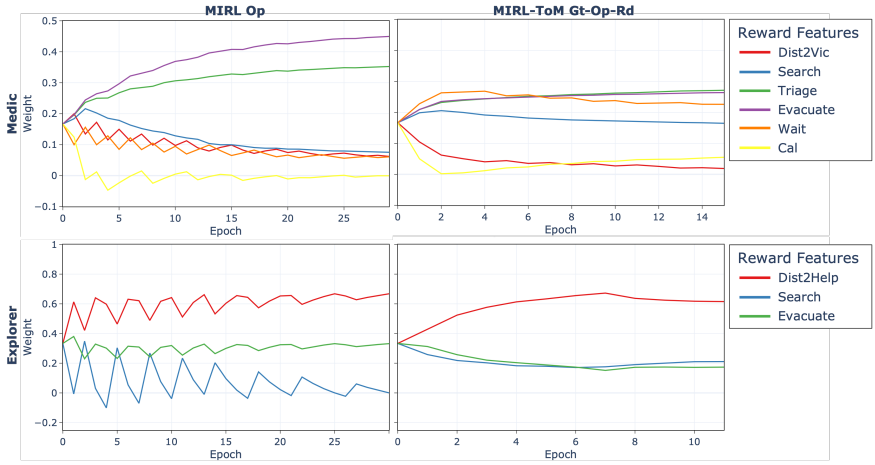


Figure 6.7: Learning curves of the reward weights during IRL for Medic (top) and Explorer (bottom) in two experimental conditions: ***cond*-2** (left) and ***cond*-3** (right).

For $cond$-3 and $cond$-4, we perform Bayesian inference using PsychSim by computing the posterior distribution over agent models given their behavior at each step of the 16 collected trajectories using Eq. 6.1. The mean model probabilities at each timestep are shown in Fig. 6.6. In $cond$-3 (Fig. 6.6 left), the Gt models can clearly be identified apart from the Rd models, meaning

that the observed behavior does denote different agent strategies (otherwise, a random behavior could be equally probable). In addition, the Op model has a very low associated probability at the end of trajectories. For $cond$-4 (Fig. 6.6 right), the Medic agent is identified as being more task-oriented, which makes sense since it has available the full action set. However, for the Explorer agent, it is harder to differentiate between the Tk and Sc models, meaning that the ground-truth behavior is perceived as being equally task- and social-oriented.

6.3.4 Decentralized MIRL Results

We applied the proposed decentralized MIRL-ToM approach in each of the aforementioned experimental conditions, with initially uniform reward weights (θ_0), maximum 30 IRL training epochs, and a learning rate (α) exponential decay of 0.9. For the forward pass (Alg. 3, line 8), we collected 16 Monte Carlo trajectories in PsychSim to estimate the FCs.

Table 6.3: Learned reward functions for the two agents, for all conditions in the known teammates setting.

Feature	**Profile θ_i**		**MIRL θ_i**		**MIRL-ToM θ_i**	
$\phi_i(s,a)$	Gt	Op	$cond$-1	$cond$-2	$cond$-3	$cond$-4
Dist2Vic	0.06	-0.06	0.02	-	0.02	0.00
Search	0.06	-0.06	0.18	-	0.16	0.16
Triage	0.19	-0.19	0.23	-	0.27	0.29
Evacuate	0.63	-0.63	0.25	-	0.26	0.28
Wait	0.03	-0.03	0.27	-	0.23	0.26
Call	0.03	-0.03	0.06	-	0.06	0.01

(a) Medic

Feature	**Profile θ_i**		**MIRL θ_i**		**MIRL-ToM θ_i**	
$\phi_i(s,a)$	Gt	Op	$cond$-1	$cond$-2	$cond$-3	$cond$-4
Dist2Help	0.25	-0.25	0.01	-	0.62	0.26
Search	0.25	-0.25	0.23	-	0.21	0.43
Evacuate	0.50	-0.50	0.76	-	0.17	0.30

(b) Explorer

The learned reward functions inferred via MIRL-ToM for each condition are summarized in Table 6.3. We observe that the reward weights obtained for $cond$-3 and 4 using ToM reasoning

are very close to those of $cond$-1, where the Gt model is used for both agents. Especially for $cond$-4, we see that model inference via ToM captured the behavior of the teammates accurately enough to allow for the recovery of the original behavior. Some results in this table can better be understood by looking at the learning curves, i.e., the evolution of the reward weights θ_i at each IRL iteration. In particular, Fig. 6.7 shows the curves for $cond$-2 and $cond$-3. As can be seen, the reward weights converge to their final values approximately within 15 training epochs except for $cond$-2, where agents always believe the other agent uses the Op model. The reward weights for this condition were not able to converge (Fig. 6.7, left), either oscillating or not converging within the maximum number of iterations.[4] Because the agent uses an incorrect model of its teammate, the resulting coordination behavior, computed via the best-response strategy learned by the agent given the current rewards and the other's model distribution (Alg. 3, line 8) leads to feature counts that are inconsistent with the empirical ones (lines 9-10), which in turn leads to a dramatic change in the reward vectors (line 11) in the next iteration.

As shown in Table 6.3, the reward weights recovered through IRL do not exactly match the ground-truth rewards (Gt column). Still, they allow us to recover the relative importance that some features have on the agents' behavior, e.g., dealing with victims (including waiting for the Explorer for evacuation) is more important for the Medic agent while searching for and evacuating victims are the priorities of the Explorer agent. As discussed earlier, different reward functions can lead to similar behavior, so a better way to investigate the similarities between the learned and expert behavior is by using similarity metrics. Here we use the following:

FC Diff: feature count difference between a policy using the IRL rewards vs. demonstrated behavior: $\sum_i |\phi_i^\theta - \phi_i^\mathcal{D}|$.

π Div: mean Jensen-Shannon Divergence [168] between the action distributions of the ground-truth and learned policy, over all steps in all trajectories: $\overline{JSD}(\pi(a_j|s_j;\theta), \pi(a_j|s_j;\theta_{gt}))$.

The results are summarized in Table 6.4. The Profile columns correspond to the best (Gt) and

[4]The seemingly converging curves are due to the decreased learning rate α.

Table 6.4: Similarity of recovered behavior for the two agents, for all conditions in the known teammates setting.

Feature	**Profile**		**MIRL**	**MIRL-ToM**	
$\phi_i(s,a)$	Gt	Op	$cond$-1	$cond$-3	$cond$-4
Empirical Feature Counts					
Dist2Vic	11.86	0.00	11.53	15.39	14.48
Search	2.40	0.00	1.75	3.13	2.06
Triage	2.97	0.00	2.91	2.97	2.94
Evacuate	2.94	0.00	2.81	2.63	2.66
Wait	8.97	0.00	10.88	6.28	7.78
Call	4.78	0.00	2.72	4.38	4.44
Similarity Metrics					
FC Diff	2.38	30.55	4.52	**7.53**	**4.50**
π Div	0.00	0.25	0.13	**0.10**	**0.13**

(a) Medic

Feature	**Profile**		**MIRL**	**MIRL-ToM**	
$\phi_i(s,a)$	Gt	Op	$cond$-1	$cond$-3	$cond$-4
Empirical Feature Counts					
Dist2Help	6.50	0.00	2.41	6.45	4.24
Search	6.59	0.00	7.06	6.34	7.38
Evacuate	2.94	0.00	2.81	2.63	2.66
Similarity Metrics					
FC Diff	2.25	14.84	3.06	**2.13**	**1.26**
π Div	0.00	0.32	0.02	**0.03**	**0.04**

(b) Explorer

worst (Op) behavior similarities achieved. The FC Diff for Gt denotes the expected error in sampling policies using the same rewards but different random seeds (hence not zero). As the number of simulated trajectories (currently 16) increases, the FC Diff would approach to zero. The MIRL column shows very similar feature counts and behavior, which is again expected because it corresponds to using a fixed Gt model of the teammate (no ToM). As for the MIRL-ToM columns, they show that our method can achieve very similar behavior (highlighted in bold), which denotes the importance of the model inference phase. Interestingly, more similar behavior was attained in $cond$-4, which does not include the Gt rewards in the model distribution, although we did not assess the statistical significance of this result. We also note that π Div is not as close to zero due

to the observed difference in the reward weights in Table 6.3, and the probabilities are dependent on the reward weight magnitude.

6.3.5 Unknown Teammates Setting

In this section, we consider demonstrations that are collected from a team of agents who form beliefs about each other's goals and update them based on observations of their behavior *during* task performance. We again use PsychSim to generate the demonstrated trajectories, but now the expert agents infer the models of each other given a uniform prior over three reward profiles: Gt, Op, and Rd. We applied MIRL-ToM by performing model inference given the demonstrations followed by MaxEnt IRL for each agent. In Table 6.5, we compare the empirical feature counts, computed from the demonstrations (column Demo) where agents use the ground-truth rewards, θ_{gt}, with the feature counts produced by the agents using the reward weights resulting from MIRL-ToM, θ_{irl}.

The results show that even in the case where the demonstrations are generated by agents that have imperfect models of their teammates, our approach can recover similar behavior. By performing model inference, we are able to model the process whereby the individuals "discover" about and adapt to their teammates' behavior.

6.4 Conclusions

We proposed an approach to Multiagent Inverse Reinforcement Learning using Theory of Mind reasoning (MIRL-ToM). Unlike other approaches to MIRL, our method does not assume full knowledge of teammates' reward functions while performing the task to compute the equilibrium strategy. Rather, we use ToM to reason about how each individual might have modeled the intentions of others by observing their behavior throughout the task, given a set of baseline reward profiles. We then break down MIRL into single-agent MaxEnt IRL for each individual, where we infer the reward functions guiding their behavior conditioned on the inferred distribution over profiles. Ascribing mental states and reward models of teammates not only accounts for im-

Table 6.5: Similarity of recovered behavior for the two agents in the unknown teammates setting.

Feature	Demo	MIRL-ToM
$\phi_i(s,a)$	θ_{gt}	θ_{irl}
Empirical Feature Counts		
Dist2Vic	14.05	14.44
Search	1.99	2.16
Triage	2.49	2.97
Evacuate	2.68	2.81
Wait	7.13	7.88
Call	5.88	5.25
Similarity Metric		
FC Diff	-	2.55

(a) Medic

Feature	Demo	MIRL-ToM
$\phi_i(s,a)$	θ_{gt}	θ_{irl}
Empirical Feature Counts		
Dist2Help	6.49	7.23
Search	8.25	6.50
Evacuate	2.68	2.81
Similarity Metric		
FC Diff	-	2.63

(b) Explorer

perfect knowledge about individual behavior and uncertain strategies of others but also enables a computationally efficient approach to compute the equilibrium strategy in a decentralized manner. The proposed MIRL-ToM shows the ability to recover similar behavior in terms of trajectory feature counts in both known- and unknown-teammate cases. In addition, although we experimented with a 2-player collaborative scenario, our framework can be applied to n-player, general-sum problems.

Our preliminary experiments were conducted using synthetic expert demonstration data to show the validity and effectiveness of MIRL-ToM. A natural next step is to apply the method to data collected from humans to provide insights into which factors—task-, social-, and emotional-related—affect their behavior the most in collaborative settings. Further, we will explore more interpretable reward structures that do not rely on linear relationships between basis features. We

are also interested in exploring ways to create the baseline profiles because the effectiveness of MIRL-ToM seems to rely heavily on the initial set. One option is to create an iterative approach, whereby the learned reward functions in one iteration are added to the set of baseline profiles until convergence.

CHAPTER 7

Bayesian Online Strategy Adaptation for Multi-Agent Teams in Open-World Novelty

With the capability of learning coordination strategies and recognizing unexpected decision behavior in the learning and interpretation phases discussed in Chapter 2 through Chapter 6, this chapter focuses on the development of adaptive coordination strategies for autonomous teammates to operate in environments that are beyond the training scenarios and the learned expertise.

7.1 Introduction

Multi-agent systems (MAS) have shown effective capabilities in automating decision-making processes and have increasingly been used in diverse application domains including robotics, logistics, and cybersecurity [77]. The complexity of formation in MAS can vary from one-on-one interactions to collaborative-competitive planning, and from homogeneous swarm intelligence to heterogeneous human-autonomy teaming. Powered by the advancements in artificial intelligence, individual autonomous agents in MAS can be equipped with their own decision models, often trained in simulation by reinforcement learning [20] or following game-theoretic strategies [169]. In response to dynamic demands, such decentralization brings advantages of efficient resource allocation and formation flexibility [128].

In real-world scenarios, however, MAS is expected to operate under open-world novelty [63]. We define such novelty as the surprising or adversarial scenarios different from the known environment and learned expertise. Without the generalization ability, problems emerge when au-

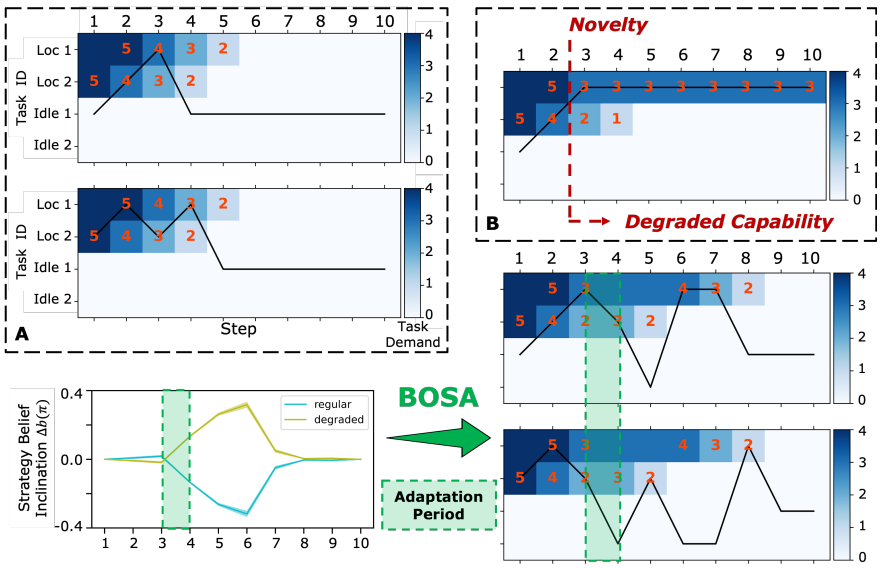


Figure 7.1: Overview of BOSA in addressing open-world novelty. (A) Successful completion in the same operation as training. (B) Failure when the capability degradation novelty is introduced. BOSA with a Bayesian strategy selection mechanism can achieve fast adaptation in novelty.

tonomous agents operate in novelty, particularly when encountering unforeseen events such as agent breakdowns [170] under fluctuating weather, unfamiliar terrain, or adversarial attacks [171]; when attempting to integrate with human teammates with different preferences [172]; when failure is not an option [173]; or when additional interventions are not available during urgent situations [174]. These challenges are especially pronounced in multi-agent settings, where even subtle disturbances might lead to catastrophic failures. Consider a multi-agent teaming scenario shown in Fig. 7.1 where three agents collaborate to apply enough extinguishing capabilities to bring down the demands of 2 fire tasks, and agents follow the policy trained by reinforcement learning [128]. With regular agent capabilities, the team is able to complete the tasks within 5 steps (Fig. 7.1A, the decisions of the first two agents). The number on top of the colored blocks indicates the total applied capabilities on each task at each step. When an unknown novelty occurs (Fig. 7.1B), the first agent in the team has a sudden capability degradation by 1, and the team can never complete the operation with the same policy. Therefore, there is a need for multi-agent systems to enable instant online strategy adaption under novelty, as otherwise these novel factors significantly prevent the systems from achieving operational synergy and effective performance.

In literature, multi-agent reinforcement learning (MARL) approaches [27, 60–62] allow agents to learn collaborative behavior to maximize operational objectives in enclosed environments, but performance cannot be guaranteed in the presence of the unknowns. Risk-takers can be modeled to address completely random and unexpected events [27], but exploratory behavior does not produce effective solutions. Inverse reinforcement learning (IRL) methods [53] learn the intrinsic reward functions from expert demonstrations as the ability of adaptation. Reward functions drive the decision-making process instead of mimicking the behavior. IRL has shown promising results in applications involving multiple agents, such as adapting behavior to unknown teammates [64], adapting a single educator to various learners [65], and adapting driving behavior to pedestrians or unfamiliar environments [66]. However, the learned reward functions do not directly identify the characteristics of novelty in the open world, and an out-of-distribution novelty could completely fail the system (e.g., unsafe driving behavior caused by an unknown object).

With the increasing need for generalizing capabilities in novel situations, research has also focused on the paradigm of open-world learning [67] —agents with limited experience revise their expertise to perform beyond known conditions. Methods shift toward online adaptation and continual learning [68, 69] for single-agent problem domains, where agents can adjust the learned decision models based on recent experiences [70] or global feedback [71]. Another category of approaches attempts to infer novelty and embed it into the decision-making process. Model-based Bayesian reinforcement learning [72] updates transition dynamics and estimates state values; a sampling-based approach formulates newly injected rules in Monopoly games [73]; failure situations initiate new learning instances [74]; and a neural-symbolic framework combines counterfactual reasoning and reinforcement learning for computer vision novelty [75]. However, in a novel scenario demanding an urgent response, these methods would still suffer from slow adaption or require additional training.

In this work, we propose Bayesian Online Strategy Adaptation (BOSA) to achieve fast adaptation by selecting the most suitable team strategy to deploy. BOSA is a three-phase approach to address novelty in multi-agent systems (i.e., capability degradation caused by unexpected events):

1) *Offline Preparation*. When agents operate in the online deployment phase, it is unrealistic to initiate another strategy training, especially under such urgency. Therefore, instead of spending time exploring the solution and adjusting the strategy to a novelty online, BOSA first collects a pool of feasible team strategies handling various novel situations and then collects the likelihood of events (e.g., unusual operation dynamics) caused by the corresponding novel cases. 2) *Online Inference*. To map novelty to strategies, BOSA adopts Bayesian reasoning based on the collected likelihood of novel events, maintains a probabilistic distribution (belief) over the collected strategies, and actively infers the most probable team strategy while perceiving online observations on possible novel events. 3) *Online Adaptation*. BOSA provides two heuristics for strategy selection, and the heuristics could significantly impact the speed of adaptation. Sometimes selecting the strategy with the most probability increase (Inclination) is more desirable than the strategy with the highest probability (Greedy), as the belief inclination is an early indication of the changes resulting from novelty. We conduct a sensitivity analysis for BOSA on a strategy belief convergence study, validating that BOSA can recognize the correct strategy to use. We then demonstrate the effectiveness of BOSA under a sudden novel event compared with noisy rational decision-making. As BOSA does not collect every novel situation and the corresponding strategy during offline preparation, we investigate whether one strategy can adapt to multiple novel situations as well as the scalability of strategy adaptation in multi-agent operations.

The contributions of this work are:

- Developing an online adaptation approach BOSA for operating in open-world novelty without re-training to infer the most suitable strategy based on offline preparation, online observations, and Bayesian reasoning,
- Proposing strategy selection heuristics (Greedy and Inclination) that achieve fast and even instant adaptation in multi-agent operations under sudden capability degradation events,
- Investigating the performance of novelty adaptation and the need for diverse strategies in different operation scales.

7.2 Methods

7.2.1 Preliminaries

The decision-making formulation and training procedure for multi-agent task allocation (MATA) problems follows our previous study discussed in discussed in Chapter 3 Section 3.2.

In the context of MATA problems, we assume that all agents in the team possess their own decision models and collaborate to achieve common team objectives. With increased intelligence, agent decision models are usually trained in a simulated environment (offline) and deployed to a real-world operation (online). This section covers an offline strategy collection by transfer learning and the proposed online strategy adaptation algorithm for novel situations using Bayesian update.

7.2.2 Offline Strategy Collection

Under our MATA formulation, it is assumed that agents always operate within the same problem domain, and we do not expect agents to acquire a completely irrelevant attribute for novelty adaptation. With reasonable novelty, unexpected events can lead to two adversarial changes to the operation. First, a sudden increase in task demands or a change in task dynamics can be triggered by a variety of novel events such as the unforeseen fire spread in multi-agent disaster relief applications, the arrival of excessive orders in manufacturing system applications, etc. With the proper formulation of the task demand state, this novelty can be addressed, as agents would have known the best action with the ability to observe task demands and realize the change in their knowledge. Second, a sudden decrease in agent capabilities can result from novel events including machine failure, natural disasters, and overestimated capabilities, which are proven detrimental to the success of operations [154].

In this study, we focus on unexpected capability degradation as the general consequence of novel situations. We refer to the trained policies of a team of agents as the team strategy π. To collect the team strategy pool, we train a baseline team strategy in a regular situation, where agents have maximum capabilities. Next, we collect multiple strategies given different novel situations

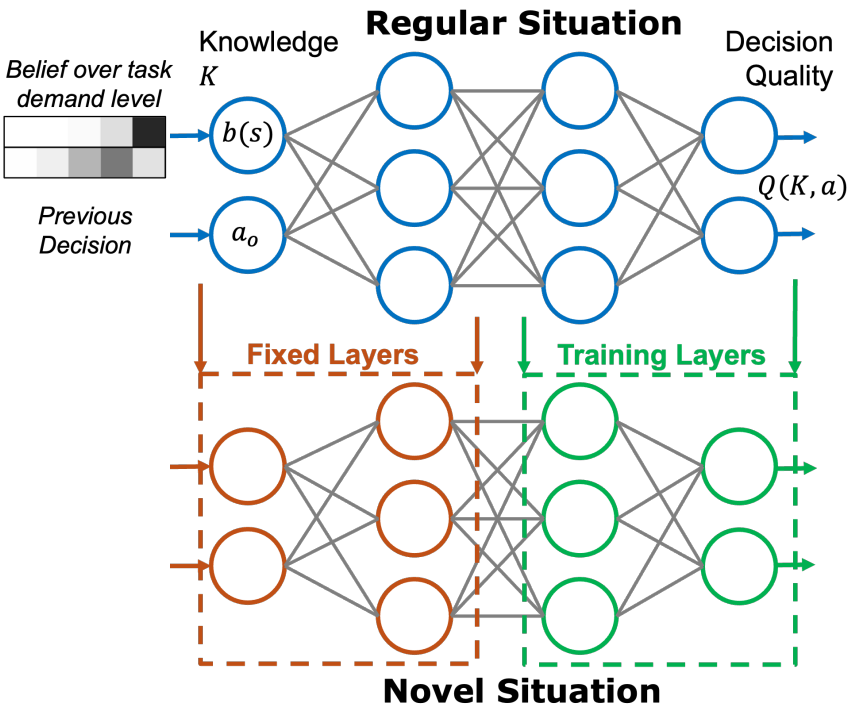


Figure 7.2: Applying transfer learning for Q-networks to obtain strategies under novel situations

under different levels of capability degradation by transfer learning [175], as illustrated in Fig. 7.2. The transfer learning approach ensures efficient strategy collection, as only a portion of the Q-network weights are trained instead of training from scratch. When obtaining the team strategy for a novel situation using transfer learning, the Q-networks of the baseline team strategy get duplicated (having the same network structure and parameters), the first couple of layers are omitted from further training, and we only fine-tune the last layers with a set of degraded agent capabilities. In addition, the number of strategies in the pool has trade-offs in adaptation performance and computational efficiency. The rule of thumb is to at least obtain a strategy trained with minimum agent capabilities to satisfy the bottom line of team performance.

7.2.3 Bayesian Online Strategy Adaptation

With a pool of team strategies trained under various novel situations $\{\pi_i\}$, we propose the Bayesian Online Strategy Adaptation (BOSA) algorithm to maintain a probability distribution over

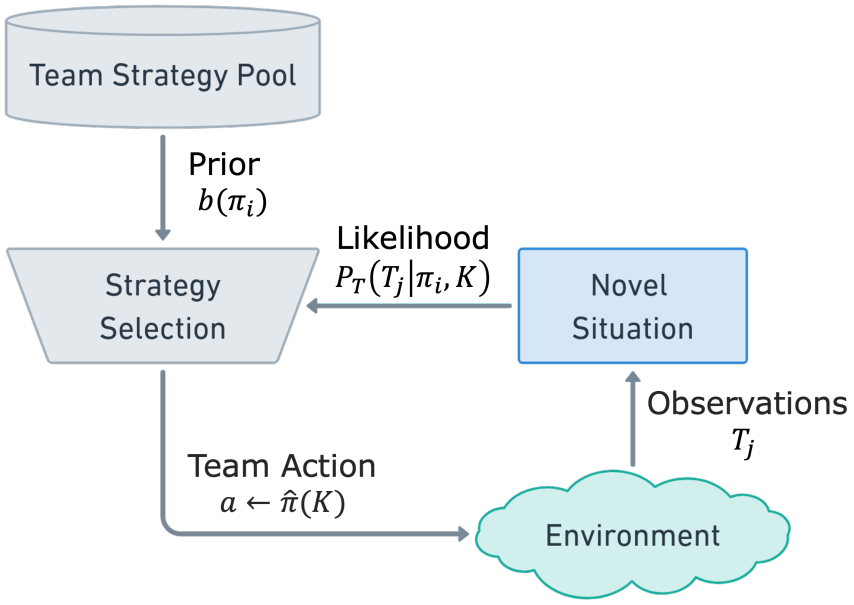


Figure 7.3: Bayesian online strategy adaptation mechanism

the team strategies $b(\pi_i)$ and to select the most suitable one while observing the task state, as shown in Fig. 7.3. BOSA requires the strategy pool and the corresponding prior belief distribution as the input (Algo. 4 Input). The prior can be a uniform distribution (i.e., the strategy is equally likely to be selected) or can be based on prior knowledge (i.e., the strategy trained with maximum capabilities is most likely to be selected assuming no capability degradation at the beginning). Next, we collect the task improvement likelihood $P_T(T_j|\pi_i, K)$ for each task j, meaning how likely the task would improve at each state K using each strategy π_i (Algo. 4, Line 1). Task improvement is directly related to the operation dynamics and agent capabilities can serve as the evidence for novel events. Note that $T_j \in \{0, 1\}$ is a binary variable, indicating whether the task demand is improving. Therefore, when the task is not improving in the online phase, some strategies become less likely to be selected (i.e., decreased probability in the belief $b(\pi_i)$ after Bayesian update).

During the online adaptation phase, the algorithm chooses a team strategy $\hat{\pi}$ based on the belief, and the team executes the actions $a \leftarrow \hat{\pi}(K)$ based on their knowledge state K. After taking the action, the team observes whether each task has improved T_j (Algo. 4, Line 3-6). With the task improvement observation T_j and the collected improvement likelihood P_T, a Bayesian update is

performed as follows on the strategy belief distribution to inform the most probable strategy to be selected for the next step:

$$b'_j(\pi_i|T_j, K) \propto P_T(T_j|\pi_i, K)b(\pi_i). \tag{7.1}$$

As the improvement likelihood is collected on each task, we have a set of updated strategy belief distributions b'_j for each task j. To aggregate them together, we can either take the average or summation over all the tasks and then normalize the belief, namely $b'(\pi_i) \propto \sum_j b'_j(\pi_i)$ (Algo. 4, Line 7-8. Here, taking the product of strategy beliefs over all tasks is also an option but is considered as too aggressive, because a single zero multiplier $b'_j(\pi_i) = 0$ would annihilate the product and completely disregard the strategy. Alternatively, one can use one variable to track the improvements of all tasks jointly (i.e., $T = (T1, \ldots, T_l)$). It provides relatively fine information for inference but is also considered aggressive, as it only relies on one piece of information. Therefore, the summation aggregation is preferred in this case. In addition, BOSA assumes that the task improvement observation is global information, the team strategy selection is a centralized decision, and agents still follow the corresponding Q-networks in that strategy to execute the tasks.

Algorithm 4 Bayesian Online Strategy Adaptation (BOSA)

Input: strategy pool $\{\pi_i\}$, prior $b(\pi_i) = b_o(\pi_i)$

1: $P_T \leftarrow$ collect improvement likelihood
2: **for** each online operation step **do**
3: $K \leftarrow$ retrieve a common knowledge state
4: $\hat{\pi} \leftarrow$ *SelectStrategy*$(b(\pi_i))$
5: $a \leftarrow \hat{\pi}(K)$
6: $T_j \leftarrow$ observe task improvement $\forall j$
7: $b'_j(\pi_i) \leftarrow$ *BayesianUpdate*$(b(\pi_i), P_T(T_j|\pi_i, K))$
8: $b'(\pi_i) \propto \sum_j b'_j(\pi_i)$
9: $b(\pi_i) \leftarrow b'(\pi_i)$
10: **end for**

7.3 Results

To analyze the proposed BOSA algorithm in multi-agent task allocation settings, we consider a collaborative fire-fighting operation scenario. The operation has a team of 3 agents, and the state

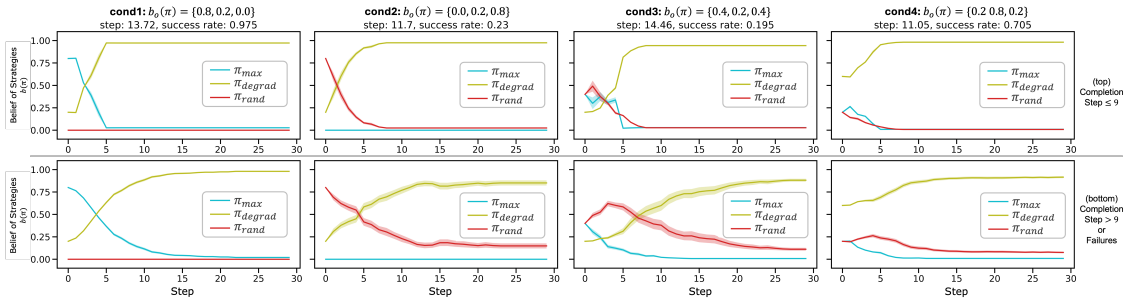


Figure 7.4: Strategy belief convergence under 4 conditions with different initial belief distributions. For each condition, we run 100 realizations and evaluate the step of completion and success rate within 30 steps. The realizations are then separated based on completion steps. Top plots show the cases that the team completes the operation with at most 9 steps; bottom plots show longer-step or failing cases. The shaded area indicates the standard deviation of the strategy belief.

of the operation is described by 2 fire task locations. The fire task demand level is discrete and in the range of $[0, 4]$. The operation starts with all fire tasks at demand level 4, and the operation completes when all fire levels reach 0. Agents decide which fire task to work on or stay idle. Each agent has a fire-fighting capability, and the capability level is also discrete and in the range of $[0, 4]$. Agents can collaborate on the same fire task, where the total level of their fire-fighting capabilities is applied to the fire task (i.e., combining power and extinguishing fire together). To model the dynamics between fire capability s_C and fire demand s_D, we assume that the fire level would decrease by 1 level with a probability of 90% if $s_C > s_D$, the fire level would increase by 1 level with a probability of 50% if $s_C < s_D$, otherwise the fire level stays the same. Fire levels cannot increase once reaching level 0. The operation might take more steps to complete if not enough capabilities are applied to prevent the fire level from increasing. In addition, a penalty will be given if agents over-use their capabilities (i.e., $s_C > s_D + 1$).

To ensure effective collaboration, agents must decide where to allocate their fire-fighting power and avoid over-usage. When two teams have different sets of fire capabilities, the team strategies are expected to be completely different. When a novelty that leads to capability degradation occurs, the team is almost unlikely to complete the operation without adapting to another strategy, recalling Fig. 7.1.

7.3.1 Convergence of Strategy Adaptation

To validate the proposed BOSA algorithm, we evaluate whether the most suitable strategy can be recognized in the operation under different initial belief conditions. For this study, we consider three trained team strategies to choose from: 1) π_{max}: three agents have the firefighting capabilities of $[4, 3, 2]$, trained by Deep Q-Learning, 2) π_{degrad}: three agents have the capabilities of $[3, 2, 1]$, trained by Deep Q-Learning, and 3) π_{rand}: three agents have the capabilities of $[4, 3, 2]$ but select actions randomly.

Next, we collect the task improvement likelihood data by first applying each strategy π_i at each possible fire task demand state s_D for $N = 1000$ realizations and then counting the number of times ($\sum_{r=1}^{r=N} T_j^r$) each task j improves. The likelihood is then $P_T(T_j|\pi_i, s_D) = (\sum_{r=1}^{r=N} T_j^r)/N$. The improvement likelihoods of 2 tasks at different states, denoted as $P_T(\pi_i)$, are shown in Table 7.1. With the Bayesian reasoning functionality, BOSA updates strategy belief based on task improvement observations and likelihoods. For example, after taking an action at state $(4, 1)$, if the team observes the new state $(4, 0)$, meaning only the second task improves $T_2 = 1$, the strategy belief will be in favor of π_{degrad} for the next step.

Task State	$P_T(\pi_{max})$	$P_T(\pi_{degrad})$	$P_T(\pi_{rand})$
(4, 4)	(0.00, 0.94)	(0.00, 0.91)	(0.07, 0.06)
(4, 1)	(0.91, 0.00)	(0.00, 0.87)	(0.07, 0.15)
(2, 3)	(0.81, 0.90)	(0.88, 0.05)	(0.20, 0.05)
(1, 1)	(0.89, 0.95)	(0.93, 0.98)	(0.09, 0.04)

Table 7.1: Collected task improvement likelihoods for 2 fire tasks using three strategies at different fire demand states.

With the three strategies mentioned and the corresponding collected likelihoods, a convergence experiment is conducted to show that the correct strategy can be inferred by BOSA, as illustrated by Fig. 7.4. In this scenario, agents have degraded capabilities $[3, 2, 1]$, and the belief of the degraded strategy $b(\pi_{degrad})$ is expected to converge to close 1.0 among the three strategies. For each step, the team selected the strategy randomly based on the belief distribution ($\hat{\pi} \sim b(\pi_i)$). We validate such convergence with 4 initial strategy belief distributions. In condition 1 and 2 (Fig. 7.4, $cond1$

and $cond2$), the team differentiates between two strategies, leaving the belief of the third strategy to be 0.0. After four steps, BOSA infers that $b(\pi_{degrad})$ has the higher probability, increased from $b_o(\pi_{degrad}) = 0.2$. In Fig. 7.4 $cond3$, BOSA recognizes the π_{degrad} among three but takes longer. In $cond4$, BOSA reassures π_{degrad} is the most suitable one. Moreover, when the operation takes longer to complete (Fig. 7.4 $cond3$ and $cond4$ bottom plots), the other two strategies follow $b(\pi_{rand}) > b(\pi_{max})$, meaning the random strategy is preferred than the one trained with maximum capabilities. This observation is reasonable because when we see tasks rarely improve, it is intuitive to take exploratory actions rather than sticking to a trained plan.

7.3.2 Novelty: Sudden Capability Degradation

In this experiment, we evaluate whether BOSA can quickly adapt strategies when a sudden novelty is introduced. Specifically, agents start the operation with the maximum (regular) capabilities $[4, 3, 2]$, and their capability levels are degraded by 1 starting at step 3. We consider two strategies for the team π_{max} and π_{degrad}, and the initial strategy belief distribution is $\{b(\pi_{max}) = 0.9, b(\pi_{degrad}) = 0.1\}$. To ensure operational effectiveness, the team does not select the strategy randomly based on the belief distribution. Instead, we propose the following two strategy selection heuristics:

1. Greedy $\hat{\pi_G} = \text{argmax}\, b(\pi_i)$: The team selects the most believed (probable) strategy to deploy.

2. Inclination $\hat{\pi_I} = \text{argmax}\, \Delta b(\pi_i)$: The team selects the strategy, of which the belief has the most increase. The belief inclination can be considered as the derivative of the strategy belief and consequently can be noisy, leading to frequent strategy switching. To prevent this, we apply a threshold filtering to only switch the strategy when the increase in belief is above 0.1.

The strategy belief $b(\pi_i)$, the strategy belief inclination $\Delta b(\pi_i)$, and the corresponding team behavior under sudden capability degradation are shown in Fig. 7.5. With the greedy strategy selection

heuristic (Fig. 7.5a), it takes 3 steps for the strategy belief to converge to a larger π_{degrad}. Such a gradual change is inherited by the nature of Bayesian update and provides a more conservative strategy adaptation. Under urgent situations, however, a faster adaptation is desired. Therefore, the inclination heuristic is preferred (Fig. 7.5a) and allows instant strategy switching (with only 1 observation) based on the strategy belief increase.

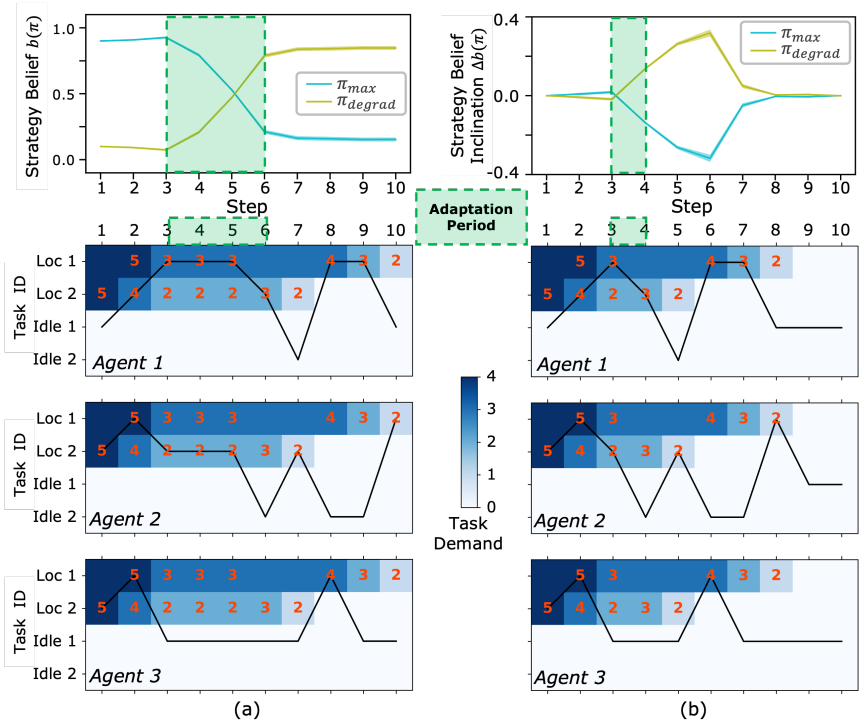


Figure 7.5: Strategy adaptation under a sudden capability degradation at step 3 with two strategy selection heuristics: (a) Greedy and (b) Inclination. Black lines indicate agent decisions over steps. Blue colors indicate the remaining task demand level for each location. The number on top of the colored cell is the total capability applied, summed over the agents selecting the same task.

In addition, we compare the performance of two BOSA strategy selection heuristics (denoted as BOSA-$\hat{\pi}_G$ and BOSA-$\hat{\pi}_I$) with the noisy rational decision-making (NRD) mechanism studied in [27] to handle unexpected events. The parameter $\phi \in [0, \inf]$ determines how rational the agents make decisions, in which an action is selected randomly based on the probability distribution: $Pr(a, *) = \frac{\exp(\phi \cdot Q(*,a))}{\sum_{a_i} \exp(\phi \cdot Q(*,a_i))}$. The noisy rational strategies are denoted as NRD-π_ϕ; when $\phi = 0$, the team makes completely random decisions, denoted as π_{rand}; when $\phi \to \infty$, the team makes completely rational decisions, which becomes π_{max}. The performance statistics of the strategies

Table 7.2: Comparison of BOSA strategy selection heuristics and noisy rational decision-making mechanisms.

Strategy	μ_{comp}	σ_{comp}	ρ_{comp}
π_{rand}	23.60	4.08	0.10
NRD-$\pi_{\phi=0.02}$	21.00	5.88	0.16
NRD-$\pi_{\phi=0.10}$	24.33	3.86	0.03
π_{max}	n.a.	n.a.	0.00
BOSA-$\hat{\pi}_G$	11.54	**1.53**	0.97
BOSA-$\hat{\pi}_I$	**9.66**	**1.53**	**1.00**

are summarized in Table 7.2, showing the average and standard deviation of the steps taken to complete the operation $\mu_{comp}, \sigma_{comp}$, and the completion success rate ρ_{comp} within 30 steps. BOSA outperforms NRD, as BOSA can leverage additional information (the strategy pool and online observations) to adapt to novelty.

7.3.3 Impact of Team Size on Strategy Adaptation

For the experiment in Sec. 7.3.2, the novelty is degrading team capabilities from $[4, 3, 2]$ to $[3, 2, 1]$. Therefore, any degradation that belongs to the interpolation region between these two capability sets (e.g., $[4, 2, 1], [3, 3, 1], [3, 2, 2]$, etc.) can be successfully addressed by two team strategies π_{max} and π_{degrad} but with a slight capability over-usage, as π_{degrad} would underestimate the team capabilities. When the degraded capability is $[4, 1, 2]$ (the capability of the second agent is outside of the interpolation region), the two team strategies might not be sufficient. The adaptability of BOSA would highly depend on the variety of the collected team strategies. To investigate how many team strategies are needed and to address the scalability concern of BOSA, this section aims to answer *"Whether a single strategy trained with one degraded capability set can cover multiple degradation cases?"*. Inspired by Mean Field Games [176] that assume a phenomenon in many-player systems where the individual player tends to have a negligible impact on the system, we hypothesize that *"As the operation size increases, a single strategy trained with the same degradation level can address more degradation cases that are different than the trained one"*.

We consider a variety of team sizes with proportional numbers of tasks, denoted by "nAmT".

Operation	N_{degrad} (N_{total})	ρ_{half}	ρ_{full}
3A2T	1	0.00	0.00
9A6T	5	0.47	0.16
12A8T	69	0.20	0.2
21A14T	100 (3431)	1.00	1.00
30A20T	100 (184755)	1.00	1.00

Table 7.3: Evaluating the effect of operation size on strategy adaptation under the same level of capability degradation. N_{degrad} is the number of degraded cases being evaluated, with N_{total} indicating the number of all possible cases. ρ_{half}, ρ_{full} measure the percentage of cases that complete half of, all of the tasks respectively.

For each team, one-third of agents have a maximum capability level of 4, 3, 2 respectively, and we collect a single strategy trained when every agent's capability is degraded by one (33.3% total capability degradation). Next, we apply the trained strategy to other degradation cases with the same degradation level (33.3%). We also assume that agents with capability level 4 can only degrade to 3, agents with capability level 3 can degrade to 2 or 1, and agents with capability level 2 can degrade to 1 or remain the same. For example, in the 3A2T operation, strategy $\pi_{[321]}$ is applied to the degradation case $[3, 1, 2]$, and $[3, 1, 2]$ is the only eligible degradation case to be evaluated under our assumptions. Table 7.3 summarizes the statistics for 5 operation sacles ranging from 3 agents to 30 agents, and the trends of completion rates ρ_{half}, ρ_{full} show that a single trained strategy can be applied to multiple other cases with acceptable performance when operation scale becomes larger, which agrees with our previously mentioned hypothesis. This investigation would improve the offline preparation phase of BOSA by effectively collecting the least amount of strategies to handle the most possible novel situations, especially for larger scale operations.

7.4 Conclusions

The motivation behind the proposed Bayesian Online Strategy Adaptation (BOSA) is to achieve prompt adaption to open-world novelty without the effort of re-training or continual learning. BOSA enables multi-agent teams to handle novelty by *offline preparing* a strategy pool, *online inferring* unusual events caused by novelty, and *online adapting* the most suitable strategy to de-

ploy. In our experiments, we showed that BOSA can successfully recognize the correct strategy to deploy through the Bayesian reasoning capability in a convergence study, and we demonstrated that BOSA allows the multi-agent teams to instantly (with one-step observation) adapt to a sudden capability degradation novelty with a strategy selection heuristic based on belief inclination. In addition, to address the scalability concern, we confirmed that, in larger operation scales, one strategy can provide acceptable performance for multiple novel cases with the same level of capability degradation.

CHAPTER 8

Multi-Agent Team Co-Design of Behavior and Composition for Operational Requirements

While strategy adaptation discussed in Chapter 7 enables autonomous teammates to handle unexpected scenarios, the team composition remains unchanged, preventing the team from achieving superior operational effectiveness. To address this challenge, simultaneous evolution of behavior and composition is needed. This chapter develops a team co-design framework, alternating between multi-objective behavior learning covered in 3 and heuristic-based composition design, to automate the development cycle of MAS.

8.1 Introduction

Multi-agent systems (MAS) have demonstrated growing effectiveness in tactical decision-making and have proliferated applications in real-world operations including emergency response [2], transportation management [3], healthcare organization [5], and conversational assistance in business [6]. Operating a team of agents and enabling the sharing of responsibilities in strategic planning and execution offer advantages such as resilience to failures [177], flexibility in handling heterogeneous tasks [27], and sustainable resource usage and maintenance [128]. However, accompanying these capabilities, MASs become even more complex and require careful design and validation during the development phase.

In the conventional development cycle of a single agent or robot, the configurations (arm lengths, degree of freedom, etc.) are pre-defined, and the behavior is then optimized for cer-

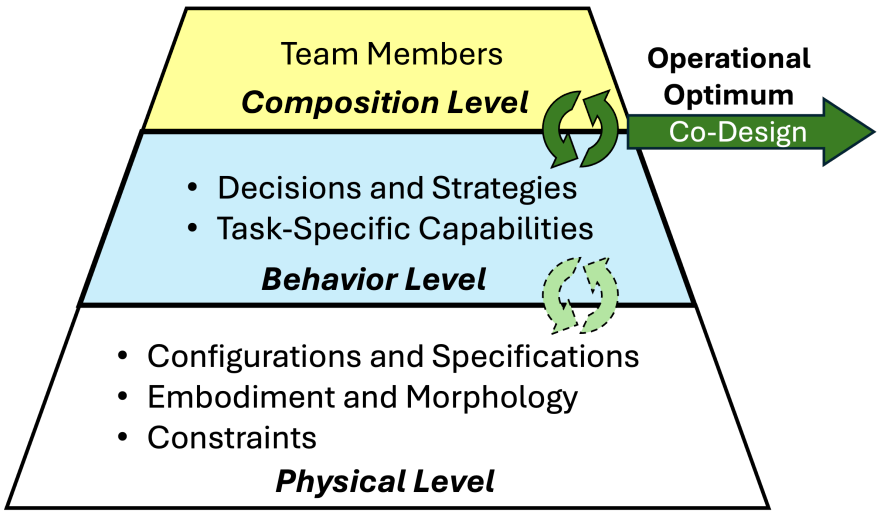


Figure 8.1: Multi-agent team co-design for operational requirements

tain tasks. When they do not perform as expected, they have to be re-configured, re-programmed, and re-evaluated [76]. This becomes a trial-and-error process and would require tremendous effort and investment. Therefore, the development process heavily relies on simulations (e.g., vehicle hardware-in-the-loop design [178] and humanoid robot design [179]), which provide the benefits of reproducibility without the need for manufacturing parts or exposing to risks in edge scenarios. For multi-agent operations, the development of a team of agents focuses not only on the individual agent configuration but also on the interaction and collaboration between them to achieve operational requirements, significantly escalating the complexity and challenges of the design process. This work aims to automate the MAS design process in simulations to generate diverse design options with a better understanding of the interdependence between operational requirements, task dynamics, agent capabilities, team composition, and strategic behavior.

We consider three design levels as shown in Fig. 8.1 when designing a multi-agent team: 1) The *Composition Level* determines the team members, i.e., how many and what types of agents belong to the team; 2) At the *Behavior Level*, team members, equipped with task-specific capabilities, make collaborative and strategic decisions to complete the operation; 3) At the *Physical Level*, the design objectives focus on how agents can be built into certain configurations (vehicle dimensions, communication devices, sensor types, etc.) under constraints by to be capable of handling

the tasks. The design processes often follow top-down or bottom-up approaches [180], but they might lead to sub-optimal or inefficient solutions. In large-scale operations, agents' decisions and capabilities directly impact whether the team can effectively satisfy the operational requirements, and the configurations at the physical level become supportive elements that could be designed and optimized in a secondary phase. Therefore, the multi-agent team design process should prioritize the development of the behavior level, where agents learn how to leverage their attributes and capabilities to make strategic and collaborative decisions, denoted as *Behavior Design*. However, with a fixed team composition, designing the behavior either through reinforcement learning [27] or mixed-integer Programming [12] still only explores the solution within the limit of the provided team composition (local optimum). For example, a homogeneous team with identical skill sets can fairly handle heterogeneous tasks requiring specialized capabilities. However, there might be a specialized team of agents that achieves superior performance. The capabilities of agents in the team composition and the decision-making of team members are inseparable. Therefore, the operational optimum can only be uncovered through the simultaneous *Co-Design* of team composition and behavior.

In this article, we seek approaches to address the fundamental problems that arise when designing multi-agent teams including understanding how team behavior provides quantitative evidence for improving team composition, how changes in team composition facilitate the overall enhancement of operational performance, and what considerations and trade-offs to make when co-evolving the team behavior and composition. In particular, we focus on the development of multi-agent teams within a multi-parameter design space for complex and dynamic operations requiring heterogeneous tasks. We tackle collaborative behavior design by multi-agent reinforcement learning with multi-objective reward functions and propose co-design frameworks that incorporate few-shots behavior learning and heuristic-based team composition design. In addition, as multiple optimums might exist in an operation, we explore the potential of extending genetic algorithms to multi-agent operations that enable the multi-agent teams to find diverse solutions using multi-team co-evolution.

This article provides the following contributions:

1. the general formulation of multi-agent operations with the incorporation of team composition, agent decision processes, and the dependency between capabilities and operational task requirements;

2. the behavior learning by interpretable multi-objective rewards considering completion efficiency, capability usage, and task switching cost;

3. the iterative co-design approach that alternates between continual behavior learning and heuristic-based composition design;

4. the co-evolutionary algorithm for finding a diverse set of optimal team compositions for multi-agent operations;

5. the investigation of team co-design in disaster-relief operations and the validation of the co-evolutionary algorithm in multi-agent multi-attribute operations.

The rest of the article is organized as follows. Sec. 8.1 introduces the research gaps in the existing literature on multi-agent operations and co-design approaches. Sec. 8.2.1 describes the preliminary formulation of multi-agent decision processes and draws the analogy with genetic algorithms. Sec. 8.2.2 formally presents the multi-agent team co-design problem. Sec. 8.2.3 introduces an Iterative Design and Learning (IDEAL) framework to solve the co-design problem and a Vote-Informed Co-Evolution (VICE) algorithm to find multiple optimal design solutions. Sec. 8.3 discusses the experiments and results, and Sec. 8.4 concludes the work with discussions on current limitations and future work. This article focuses on designing multi-agent systems (MAS) for multi-agent task allocation (MATA) operations. According to the complexity definition of taxonomy [8], our MATA operation formulation belongs to the category CD-[ST-MR-TA] (complex dependencies, single-task robots, multi-robot tasks, time-extended assignment) that involves both task assignment and scheduling, and tasks have multi-level subtasks with conditional dependencies on prerequisite tasks and applied capabilities. Designing a MAS requires the consideration

of eight features [181]. *Agreement Parameters* define the operational requirements as the common goal of a MAS, and the rest can also be categorized into the three design levels illustrated in Fig. 8.1: 1) Composition Level: *Leadership* and *Topology* define the hierarchy and organization of the agents [182]. Our MATA formulation allows team members to operate in a decentralized manner and make decisions simultaneously without following a leader or a pre-defined rule, which provides flexibility when designing the team composition. 2) Behavior Level: *Decision Function* describes the reasoning process of agents in MAS, *Heterogeneity* refers to the difference in agent characteristics or capabilities of executing tasks, the *Mobility* states whether agents are static or mobile. In our formulation, each agent in the team has a list of heterogeneous task-specific capabilities and makes strategic decisions in a dynamic setting. 3) Physical Level: *Data Transmission Frequency* and *Delay Consideration* belong to the sensor and communication configurations of agents. Our work assumes agents make task allocation decisions less frequently than real-time operating (e.g., driving), and therefore it is reasonable to have all-to-all communication without delays at every decision step. The rest of the section covers the related works that address team composition design (i.e., agent selection) or team behavior design (i.e., the design of capabilities or algorithms) separately (Sec. 8.1.1) and co-design problems for single agents and multi-agent teams in Sec. 8.1.2.

8.1.1 Composition Design and Behavior Design

In MAS, team composition design determines whether to include agents in the team given their utilities and performance metrics in an operation. Some works have studied cooperative team composition for coverage problems by gradient optimization [78] and distributing strategies to dynamic team composition through a coach-player framework [79]. Another line of works [80–82] deploys evolutionary computing to find team composition for cooperative tasks and find experts from social networks [83]. Due to the complexity of MAS, maintaining a population of teams becomes intractable, the concept of competitive and cooperative co-evolution [84, 85] is introduced to MAS that uses comparative information to modify team composition in multi-objective optimization ap-

plications including swarm intelligence [86]. However, these works assume that agent behavior does not change (e.g., fixed controllers and pre-defined rules) over generations, and they do not utilize the dependency between design the control for operational optimums.

Behavior design in MATA operations refers to being capable of making decisions and taking actions given observations or knowledge of the tasks [9, 77]. In heterogeneous teams where capabilities are different in terms of their abilities to handle tasks, MATA problems can be solved using mixed-integer programming approaches [12–14], hierarchical planning [17, 18], or genetic algorithms [19], but the agents themselves do not have their own decision-making processes. Instead, decentralized partially observable Markov decision processes (Dec-POMDPs) [20] are often used to formulate MATA operations, allowing agents to gain intelligence by reinforcement learning (RL) through interaction with dynamic environments for predator-prey operations [22] and disaster-relief operations [27], considering execution uncertainties [183] and path planning capabilities [184]. Though some works focus on resilience to adverse environments [185, 186], adaptation to changes in team composition [15, 187], or robustness when facing uncertainties in tasks and agent capabilities [12], the underlying team composition is given as the input to MATA operations, and the question on whether the team composition can reach optimal performance in an operation is not answered. As team composition is inherently connected to team behavior, our co-design framework considers both design levels simultaneously, connecting them through behavior evaluation and composition adjustment heuristics.

8.1.2 Multi-Agent Co-Design

In developing a single agent or robot, jointly evolving morphological structure and learning control policies by RL [87] can find a version of the design that is better suited for the task by understanding the interactions between learning, evolution, and the environment [88]. During evolution, the agent design either follows a distribution [89], modifies based on the current design [88], or evolves at a slower timescale than behavior to ensure convergence [90]. Works also consider multi-objective optimization to design robotic arms for motion planning tasks [91],

design unmanned aerial vehicles [92], and find a versatile design that can address multiple task scenarios [188, 189]. Inspired by single-agent co-design, approaches for MAS leveraging the evolution of team composition can drive team behavior to achieve optimal performance in operations. With complex interaction between agents, co-design for MAS is more challenging because a slight change in composition could lead to a completely different behavior. In control problems, co-design for heterogeneous teams also uses evolutionary computing [93] that puts the parameters of both team composition and agent controllers into evolution. However, to the best of our knowledge, there is limited work in multi-agent team co-design problems for MATA operations, where agents have task-specific attributes and perform operational strategic reasoning instead of control actions.

8.2 Methods

8.2.1 Multi-Agent Task Allocation

This section covers the context of multi-agent task allocation (MATA) problems in Sec. 8.2.1.1 with the formulation of agent decision processes, we illustrate the training process of agent decision models in Sec. 8.2.1.2, and we draw an analogy between agents in multi-agent systems and chromosomes in genetic algorithms in Sec. 8.2.1.3 to motivate the team composition design.

8.2.1.1 Coupling Decision-Making and Capabilities

Multi-Agent Partially Observable Markov Decision Process (MA-POMDP) [167] is a framework enabling a team of agents to collaborate and plan under uncertainty. At every time step, agents take individual actions and receive individual observations. We assume that all agents share their observations through a communication channel, allowing them to have centralized information without costs or delays. However, all agents in the team are equipped with their own decision models and act in a decentralized manner, which offers benefits in modeling individual preferences and avoids having a single joint but intractable decision model. In addition, to incorporate

heterogeneous tasks and agents with heterogeneous capabilities, we formulate the MATA operation as an extension of MA-POMDP (MATA-POMDP), which explicitly models the dynamic dependency between agent capabilities and task demands. The MATA-POMDP is defined as a tuple $(C, G, A, S_C, S_D, T, Z, O, R)$, where

- $C := \times_i C_i$ is a set of team compositions, described by the capabilities of each agent $c_i \in C_i$, where i is the index of a set of n agents, and $c_i = [c_i^1, \ldots, c_i^j, \ldots, c_i^p]$ is a list of p attributes with capability levels c_i^j,
- $G := \{g_1, \ldots, g_p, \bar{g}_1, \ldots, \bar{g}_q\}$ is a finite set of p tasks and q idle locations,
- $A := \times_i G, a \in A$, is a set of team actions composed of the decision of each agent i, executing a task or idling at a location,
- $S_C := \times_j S_C^j$, is a set of team capability states for each task $j \in \{1, \ldots, p\}$, and is mapped from the team composition by a team action $f_C : C \times A \to S_C$,
- $S_D := \times_j S_D^j$ is a set of task states described by the demand level of each task j,
- $T := Pr(s'_D | s_D, s_C, a)$ is the probability of transitioning to task state s'_D from the current state s_D with the team capability state $s_C = f_C(c, a)$ given the team action $a \in A$ and composition $c \in C$,
- $Z := \times_i Z_i, z \in Z$, is the set of team observations composed of each agent's perceived task state,
- $O := Pr(z | s'_D, s_C, a)$ is the observation probability,
- $R := R(s_D, s_C, a)$ is the reward function, indicating the cost of applying capabilities at a demand level.

The objective of the MATA operation is to find an optimal team policy π^* that maximizes the operation performance value $V(\pi) = \mathbb{E}[\sum_{t=0}^{t=\infty} \gamma^t R(s_D^t, s_C^t, a^t) | \pi, b^0]$, where γ is a discount

factor. b^0 is the initial belief (probability distribution) on the task demand states to accommodate partial observations. At every step, the belief is updated by the Bayes' Rule with team actions and observations:

$$b'(s'_D) \propto O(z|s'_D, s_C, a) \sum_{s_D} T(s'_D|s_D, s_C, a) b(s_D) \tag{8.1}$$

8.2.1.2 Information Representation and Policy Training

Adopting from our previous work [128], we use an information state $I = [a_o, b(s_D)]$ to represent the knowledge for decision-making, which is concatenated by the previous action a_o and the belief on task demands $b(s_D)$. At each step, the current action becomes the next previous action $a'_o = a$, and the belief is updated using Eq. 8.1. Here, $b(s_D)$ is the centralized belief used by all agents in the team, and a_o tracks the previous action of each agent i. The information flow and the training process are demonstrated in Fig. 8.3a.

Each agent's decision model is represented by a deep Q-network $Q(I, a; \theta)$ approximating the decision Q-value, defined as the expected total reward at the given information state I by taking an action a. We omit the agent index, as all agents have the same information representation format. To train the agent policy, we adopt Deep Q-Learning [95] for multi-agent operations. At each step, agents take actions a, apply capabilities s_C communicate observations z, incur rewards r, and update information I'. For each agent, the decision model is trained with the transition data $< I, a, I', r >$ according to the loss function:

$$Loss(\theta) = \mathbb{E}[r + \gamma \text{max} Q(I', *; \hat{\theta}) - Q(I, a; \theta)]^2, \tag{8.2}$$

where $Q(*; \hat{\theta})$ is the target Q-network updated less frequently than the decision Q-network $Q(*; \theta)$ for convergence. The action to take and the optimal policy can then be retrieved by $a = \pi(I) = \text{argmax}\, Q(I, a; \theta)$. The Q-networks of all agents are trained simultaneously.

8.2.1.3 Genetic Algorithm Analogy

In our MATA-POMDP formulation, each agent in the team is composed of a list of attributes, where each attribute describes the agent's capability level to handle a specific task. This structure mirrors how a chromosome is represented by a list of genes, as shown in Fig. 8.2. In the genetic algorithm, an offspring chromosome is generated by two parents selected from the population based on fitness. Similarly, an offspring agent can be evolved from two parents $M = \{M_1, M_2\}$ with *Crossover* and *Mutation* processes. We refer to the offspring generation as a function *CrossoverMutation*(M, p_c, p_m), where p_c indicates the probability of crossing over an attribute from either parent and p_m is the probability of mutating an attribute. The detail of how genetic algorithms are applied to multi-agent team composition design is discussed in Sec. 8.2.3.2.

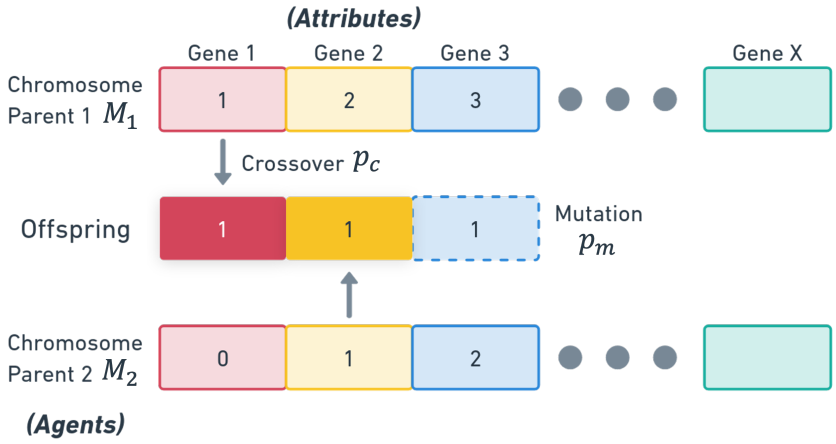


Figure 8.2: Crossover and mutation phases in conventional Genetic Algorithms with an analogy in attributes of agents.

8.2.2 Multi-Agent Team Co-Design

8.2.2.1 Behavior Design by Multi-Objective Rewards

A carefully designed reward function is crucial during the learning process for shaping collaborative behavior to achieve effective operation. We define the reward function for each individual

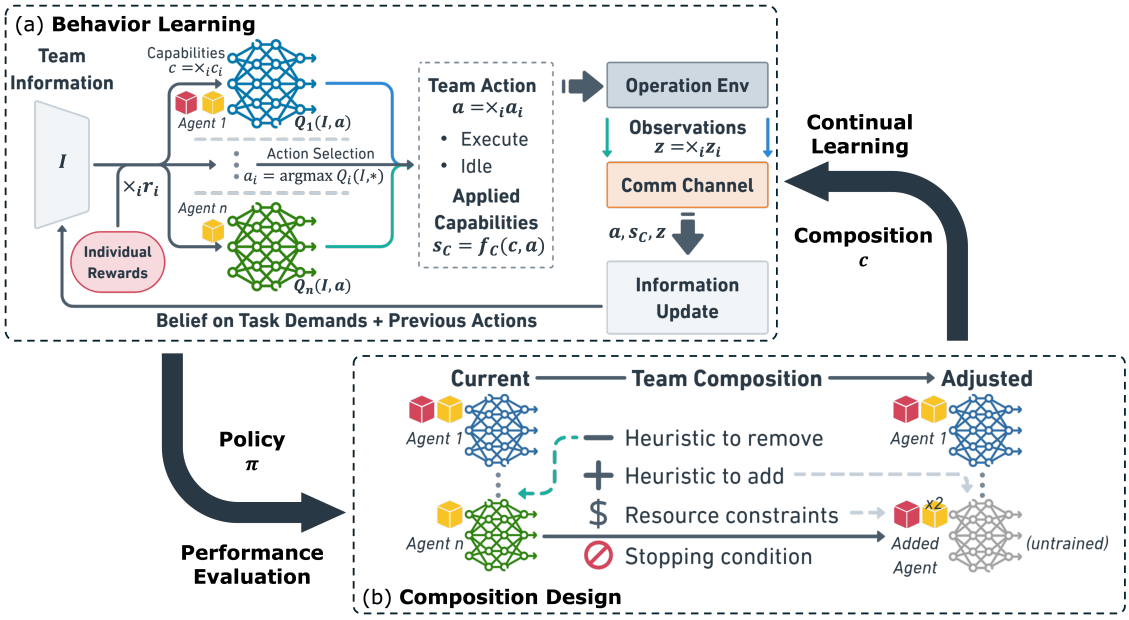


Figure 8.3: A co-design framework by Iteration Design And Learning (IDEAL). Starting at a given team composition, module (a) *Behavior Learning* performs multi-agent reinforcement learning with centralized information on tasks and capabilities applied, individual rewards, and decentralized decision models. With an evaluation procedure and performance metrics on learned behavior, module (b) *Composition Design* determines whether to remove or add agents based on constraints. With the adjusted team composition, the iterative process continues until the drop performance or no change in team composition.

agent as:

$$\hat{s}_D = \operatorname{argmax} b(s_D), \tag{8.3}$$

$$\begin{aligned} R(I, a) &= R[\hat{s}_D, a_o, a, s_C = f_C(c, a)] \\ &= R_D(\hat{s}_D) + \omega_r R_A(a_o, a) + \omega_r R_C(s_C, \hat{s}_D). \end{aligned} \tag{8.4}$$

Eq. 8.3 maps the belief space back to the task state space by considering the most believed (probable) demand level for each task. Eq. 8.4 defines the multi-objective rewards with two objectives: 1) the primary objective is operation completion. $R_D(\hat{s}_D) = -\sum_j \hat{s}_D^j$ is the penalty (negative reward) for the total remaining demands. 2) the secondary objective has a re-allocation penalty R_A and an unnecessary capability usage penalty R_C, and they are weighted by $\omega_r \ll 1$ to avoid compromising the primary objective on operation completion. Note that the primary objective is

on the team scale as all agents share the information on $b(s_D)$ to ensure they complete all tasks, and the secondary objective allows agents to have different behaviors according to their preferences.

8.2.2.2 A Co-Design Problem for Operational Requirements

With a fixed team composition c, the desired team behavior can be π trained by Deep Q-Learning and the reward shaping described in Sec. 8.2.1.2 and Sec. 8.2.2.1 respectively. We call this process *Behavior Learning* as shown in Fig. 8.3a. However, the learned behavior is considered optimal only for the given team composition, and there might exist another pair of team behavior and team composition that can push for even better operational performance. We aim to find such a combination of $(\pi, c)^* = \text{argmax}\, \mathbb{E}[\sum_{t=0}^{t=\infty} \gamma^t R(s_D^t, s_C^t, a^t)|\pi, c]$ and refer to this problem as the *Multi-Agent Team Co-Design of Behavior and Composition*.

8.2.3 Iterative and Evolutionary Co-Design

This section introduces a heuristic-based co-design approach alternating between behavior learning and composition design in Sec. 8.2.3.1 and proposes a co-evolutionary algorithm in Sec. 8.2.3.2 that evolves multiple teams simultaneously to find diverse operational optimums.

8.2.3.1 Iterative Design and Learning

The multi-agent co-design problem has two design objectives: team behavior and team composition. To solve both design problems simultaneously, we propose a heuristic-based co-design framework **I**terative **DE**sign **A**nd **L**earning (IDEAL) which iteratively performs continual learning for team behavior and team composition adjustment. During each co-design iteration, IDEAL starts behavior training for a given team composition (Fig. 8.3a), evaluates the team performance on operation completion and task improvement, and measures the individual behavior on capability usage. These metrics are important information to guide team composition design. We present the notations used in performance evaluation and the design process in Table 8.1. Next, IDEAL starts the heuristic-based team composition adjustment for continual learning, as shown in Fig. 8.3b,

based on the following heuristics:

Table 8.1: Notations Used in Performance Evaluation

	Definition
i, n	index for n agents
j, p	index for p tasks
t, H	index for H step horizons
e, E	index for E evaluation episodes
a_i^t	$a_i^t = \pi_i(I_i^t)$; decision of agent i at step t, following the trained policy π_i at the information state I_i^t
c_i^j	capability level of agent i for task j
Agent Utility Related	
U_i	agent utility
ω^j	utility weight for each task j
u_i^j	average total capability usage for task j over E episodes
$\mathbb{1}(*)$	indicator function; $= 1$ when agent i decides to execute task j at step t; $= 0$ otherwise
ζ_U	utility threshold
Capability Gap Related	
c_+^j	the capability level of each task j to be composed into a new agent
l_e^j	the capability gap to handle each task j in each episode e, inferred by f_g using the demand-capability trajectories $\{\hat{s}_D^t, s_C^t\}$ in each episode

Removing an Existing Agent

An agent is considered as removable when they do not apply their capabilities or keep idle. We quantify the agent utility U_i based on capability usage u_i^j for each task j following Eq. 8.5 and then remove the ones with utility below a threshold ($U_i < \zeta_U$). When all agents are above the threshold, we attempt to remove the one with the least utility ($c^- = \text{argmin}\{U_i\}$).

$$U_i = \sum_{j=1}^{p} \omega^j u_i^j, \qquad u_i^j = \frac{1}{E} \sum_{1}^{E} \sum_{t=1}^{H} c_i^j \mathbb{1}(a_i^t = j). \tag{8.5}$$

Adding an agent

An agent should be added when the team does not complete the operation as desired or does not have enough capabilities to efficiently satisfy task demands. With Eq. 8.6, we quantify the capability gap l_e^j based on state-action histories (demand-capability trajectories $\{s_D^t, s_C^t\}$) and identify the most needed capability level of all tasks c_+^j for the new agent c^+. Resource budget constraints are then applied $c^+ = f_b(\{c_+^j\})$ to avoid exceeding the team or individual capacity. f_g, f_b are the design functions requiring the designer's domain knowledge.

$$c_+^j = \max\{l_e^j\}, \qquad l_e^j = f_g(\{s_D^t, s_C^t\}_e). \tag{8.6}$$

Stopping conditions

We terminate the iterative process when there is no change in the team composition ($c^- \equiv c^+$) or when there is a significant decline in performance in terms of the number of steps taken to complete all tasks.

The proposed IDEAL framework can provide the following advantages for multi-agent team co-design. 1) Each adjustment can be justified by the design heuristics. 2) The iterative process allows designers to track design options and to conduct secondary trade-off analysis on these options (e.g., a team with fewer agents and lower performance might be preferred as it costs less to build them in the real world). 3) With the designer's domain knowledge, IDEAL can accommodate the designer's feedback on the operational requirements and the changes in the environment.

8.2.3.2 Vote-Informed Co-Evolution

Applying conventional genetic algorithms to multi-agent teams poses challenges in scalability, as maintaining a population of teams of agents could easily become intractable. In addition, there might exist multiple optimal solutions to the multi-agent co-design problem with two design objectives, similar to the Pareto front of a multi-objective optimization problem. To address these,

we propose an evolutionary algorithm **V**ote-**I**nformed **C**o-**E**volution (VICE) that treats each team itself as a population of agents, co-evolves multiple teams together, and records agent votes in each team to attract the best agents to the best team.

We present VICE for 3-team co-evolution in Algo. 5. Each multi-agent team i is initialized with a team composition D_i and a memory of agent votes $V_i = \emptyset$. Each team has a fixed size of N. The team composition is described by the capability set $c_j = \times_k c_j^k$ of each agent j, where k denotes each of the agent's attributes. Recall that a multi-agent team setup is analogous to a population of chromosomes c_j, and the capability level of each attribute is equivalent to a gene c_j^k in the chromosome. The 3-team format of VICE could provide comparative information of different teams, allowing them to attract team members. For each evolution, VICE follows three phases:

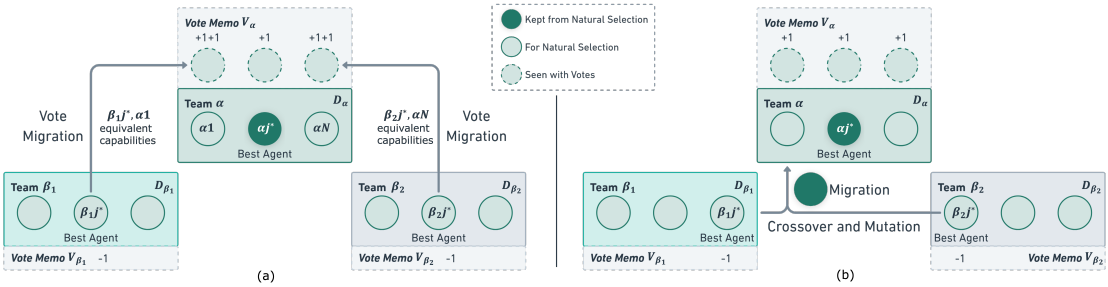


Figure 8.4: Two cases in the migration phase for Vote-Informed Co-Evolution (VICE) to attract the best agents from other teams to the best team with the voting mechanism: (a) when an equivalent agent exists in the best team, up-vote and (b) otherwise, generate an offspring for the best team.

- *Evaluation* (Algo. 5 Line 3-5). We evaluate the team performance score TS_i and individual agent performance score AS_i^j and give $+1$ votes to all agents in the best team $\alpha = \text{argmax}\{TS_i\}$ in the vote memory V_α. We then keep the best agent αj^* in the best team from the natural selection phase.

- *Migration* (Algo. 5 Line 7-15). This phase aims to attract the best agents from the other 2 teams, denoted as $\beta_1 j^*, \beta_2 j^*$, and they get -1 votes in the corresponding vote memories V_{β_1}, V_{β_2}. There are two migration cases demonstrated in Fig. 8.4. When there is an equivalent

Algorithm 5 Vote-Informed Co-Evolution (VICE)

Input: 3 multi-agent teams, each with team composition $\{D_i\}$ and vote memory $\{V_i = \emptyset\}; |D_i| = N$

1: **for** each evolution iteration **do**
2: *// Evaluation Phase*
3: $\{TS_i\}, \{AS_i^j\} \leftarrow$ *Evaluate* teams $\{i\}$ and agents $\{ij\}$
4: $V_\alpha \leftarrow$ *UpVote* agents in the best team α
5: $\alpha j^* \leftarrow$ *Keep* the best agent j^* in the best team α
6: *// Migration Phase*
7: $\beta_1 j^*, \beta_2 j^* \leftarrow$ The best agents in the other two teams
8: $V_{\beta_1}, V_{\beta_2} \leftarrow$ *DownVote* $\beta_1 j^*, \beta_2 j^*$
9: **if** Case-1 $\beta_1 j^*, \beta_2 j^* \in D_\alpha$ **then**
10: $V_\alpha \leftarrow$ *UpVote* them in the best team α
11: **else if** Case-2 $\beta_1 j^*, \beta_2 j^* \notin D_\alpha$ **then**
12: $M \leftarrow \beta_1 j^*, \beta_2 j^*$ become parents
13: offspring $\leftarrow$ *CrossoverMutation*(M, p_c, p_m)
14: $\alpha j^\# \leftarrow$ *Keep* the offspring in the best team α
15: **end if**
16: *// Natural Selection Phase*
17: **for** each agent $ij \neq \alpha j^*, \alpha j^\#$ in team i **do**
18: $M \leftarrow$ *SelectParents*(V_i)
19: $D_{ij} \leftarrow$ *CrossoverMutation*(M, p_c, p_m)
20: **end for**
21: **end for**
22: **return** Top K evolved teams seen by the 3 teams

agent in the best team (Fig. 8.4a), we give an additional $+1$ vote (migrated) for that agent in the memory of the best team V_α; Otherwise (Fig. 8.4b), the two best agents become the parents and generate an offspring agent $\alpha_\#$ to be migrated to the best team and be kept from further natural selection.

- *Natural Selection* (Algo. 5 Line 17-20). For the rest of the agents, we perform crossover and mutation with probabilities p_c, p_m respectively, where the parents are selected randomly with probabilities weighted by votes in memories instead of by individual agent scores.

Through evolution iterations, each team can become the best team and can accumulate votes for the agents that collaborate well with others in the team. With this multi-team co-evolution format, we expect to find a diverse set of optimal team compositions.

As multi-agent operations scale up (i.e., large numbers of attributes and agents in the team), the 3-team format VICE can be easily extended to a larger scale. In a general $(2k+1)$-team format, we keep the top k agents in the best team α, and the other $2k$ teams form k parent pairs for the migration phase. We assume $2k < N$ to ensure that enough agents in the team can be evolved by natural selection.

VICE is presented as an evolutionary algorithm for team composition design problems. However, it also works with the multi-agent team co-design problem by embedding the continual learning mechanism into the evaluation phase and storing the updated agent decision models into vote memories. Therefore, the agents can be simultaneously trained and evolved.

8.3 Experiment and Results

We simulate a multi-agent disaster relief operation scenario for our experiments, as shown in Fig. 8.5. In this operation, there are 2 disaster locations and 2 types of tasks at each location: extinguishing fire and rescuing victims from the burning buildings. Agents are trained to decide which task to execute with the option of staying idle. The operation has the following characteristics and assumptions:

- **Demands and Capabilities.** The task demand level s_D^j is within the range of $[0, 4]$. When the operation begins, all tasks have demand levels of 4, and reaching demand levels of 0 completes the operation. The operation has 5 agents in total, and each agent has two task-specific attributes: extinguishing and rescuing. We assume that agent capabilities can be summed together when they decide to execute the same task, and the team capability level for each task s_C^j is within the range of $[0, 5]$.

- **Task Dynamics.** A higher capability would reduce the demand with a higher probability, and demands do not increase after reaching 0. For fire extinguishing tasks, we assume that the fire demand would decrease by 1 level with a probability of 90% if $s_C^{ext} > s_D^{ext}$, stay at the same level when if $s_C^{ext} = s_D^{ext}$, and increase by 1 level with a probability of 10% if $s_C^{ext} < s_D^{ext}$.

For rescuing tasks, we assume that the rescue demand would decrease deterministically by the amount of the applied rescuing capability of the team (i.e., $s_D^{'res} = s_D^{res} - s_C^{res}$).

- **Task Dependency.** The operation also incorporates the dependency between different tasks at the same location. Specifically, when the fire level is high ($s_D^{ext} > 2$), agents cannot successfully rescue victims (i.e., $s_D^{'res} = s_D^{'res}$ with a 100% probability).

- **Behavior Expectation.** With a properly designed multi-objective reward function, proposed in Sec. 8.2.2.1 Eq. 8.4, agents would collaborate and reduce the task demands efficiently (primary objective R_D) by co-executing the same task or simultaneously executing different tasks. Over-usage of capabilities refers to the situations when the team applied more than needed capabilities to task demands (e.g, $s_C^{ext} \gg s_D^{ext}$, $s_C^{res} > s_D^{res}$, or $s_C^{res} > 0$ if $s_D^{ext} > 2$). To avoid over-usage (secondary objective R_C), agents induce idling behavior, allowing them to stand by and save the inherent operational cost. In addition, we model the penalty when agents switch tasks or change locations (secondary objective R_A), allowing them to plan routes and effectively save the reconfiguration cost.



Figure 8.5: An illustration of a disaster-relief operation. The team of agents equipped with task-specific attributes described by capabilities decides which task to execute under dynamic task demands and uncertainties.

Table 8.2: Performance Evaluation for Two Team Compositions

Metrics*	Homogeneous Team					
	$\omega_r = 0$	$\omega_r = 0.5$✓	$\omega_r = 0.75$	$\omega_r = 1.0$	$\omega_r = 2.0$	$\omega_r = 3.0$
$\rho_{H=10}$	0.980 →	0.985 →	0.980 →	0.900 ↘	0.770 ↘	0.420
$\mu_F(\sigma_F)$	6.69 (0.87) →	6.68 (0.90) →	6.73 (0.84) ↗	7.63 (0.73) ↗	8.50 (0.50) ↗	9.00 (0.00)
$\mu_C(\sigma_C)$	80.1 (1.74) ↘	47.0 (4.73) →	44.6 (6.00) →	46.1 (4.55) →	46.8 (3.60) →	47.1 (2.93)
$\mu_I(\sigma_I)$	0.68 (1.00) ↗	23.1 (2.30) →	23.6 (3.11) →	23.0 (2.22) →	24.0 (1.83) →	24.5 (1.44)
$\mu_S(\sigma_S)$	20.5 (2.99) ↘	13.5 (1.15) ↗	14.5 (1.58) ↗	19.0 (0.43) ↘	16.3 (0.84) ↘	13.0 (0.99)
	Heterogeneous Team					
	$\omega_r = 0$	$\omega_r = 0.5$	$\omega_r = 0.75$✓	$\omega_r = 1.0$	$\omega_r = 2.0$	$\omega_r = 3.0$
$\rho_{H=10}$	0.995 →	0.995 ↘	0.970 →	0.955 ↘	0.890 ↘	0.360
$\mu_F(\sigma_F)$	5.84 (0.95) →	5.82 (0.87) →	5.76 (0.82) ↗	6.88 (0.90) ↗	7.73 (0.72) ↗	9.00 (0.00)
$\mu_C(\sigma_C)$	80.8 (3.50) ↘	43.3 (5.04) ↘	39.7 (4.50) →	40.0 (4.65) →	39.7 (3.65) →	38.9 (2.47)
$\mu_I(\sigma_I)$	0.04 (0.33) ↗	29.1 (2.29) →	29.7 (1.68) →	29.3 (1.92) →	28.5 (1.55) →	29.8 (1.07)
$\mu_S(\sigma_S)$	16.8 (2.23) ↘	14.0 (0.74) ↘	12.2 (0.86) ↗	15.9 (0.82) ↗	17.2 (0.70) ↘	12.1 (0.31)

*All metrics are evaluated by following the policies trained with various reward weights ω_r. In 200 realizations. $\rho_{H=10}$ is the success rate with 10 steps, μ_F is the average number of steps to complete all tasks, μ_C is the average total capability usage, μ_C is the average number of idle decisions, μ_S is the average total number of reconfigurations, and (σ_*) is the standard deviation of the metric. The ↗, ↘ indicate the significant changes in metrics as ω_r increases.

8.3.1 Multi-Objective Rewards

8.3.1.1 Experiment Setup

To validate the effectiveness of the multi-objective reward, we consider two team compositions (5-agent team): a) a homogeneous team, where each agent has the same capabilities $[c_i^{ext}, c_i^{res}] = [2, 1], \forall i$; b) a heterogeneous team, with the composition $\left[\begin{array}{c} \boldsymbol{c}^{ext} \\ \boldsymbol{c}^{res} \end{array}\right] = \left[\begin{array}{c|c|c|c|c} 3 & 3 & 2 & 1 & 0 \\ 1 & 2 & 0 & 1 & 1 \end{array}\right]$. We give a preventive capability over-usage penalty -50 for all unnecessary cases when $s_C^{ext} - s_D^{ext} >= 2$, $s_C^{res} > s_D^{res}$ (over-usage of extinguishing and rescuing capabilities), $s_C^j > 0$ if $s_D^j = 0$, and $s_C^{res} > 0$ if $s_D^{ext} > 2$ (execution not needed). We give an allowable capability over-usage penalty $-2 * \omega_r$ in all other cases for effective usage of resources. In addition, we give a penalty $-0.5 * \omega_r$ when agents switch tasks or locations (reconfigurations) to further refine the team behavior.

8.3.1.2 Results and Discussion

We perform the Deep Q-Learning described in Sec. 8.2.1.2 for two team compositions with different weights on secondary objectives (ω_r). When $\omega_r = 0$, the problem reduces to a single objective, which is used in [27] and is the baseline we compare to.

Fig. 8.6 shows the training curves for the heterogeneous team in 6000 episodes. The homo-

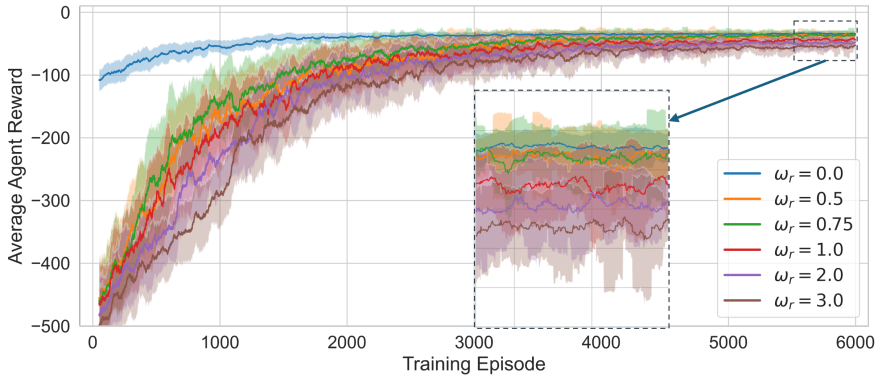


Figure 8.6: Training curves of average agent reward for the heterogeneous team composition with multi-objective reward weight ω_r ranging from 0 to 3.0. Shaded colors indicate the moving standard deviations of the training rewards in the previous 30 episodes.

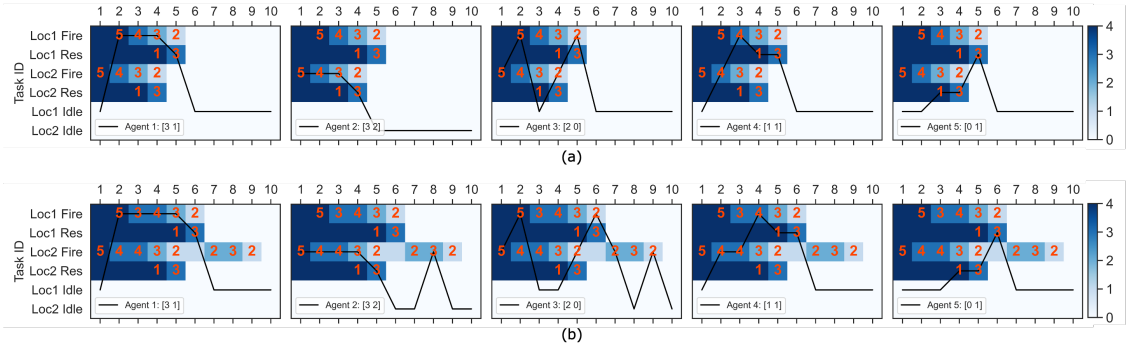


Figure 8.7: The behavior of the heterogeneous team in (a) an optimistic and (b) a pessimistic case under uncertainties. Black lines are agent decisions over steps, blue cells show the remaining task demand levels, and the red numbers indicate the capabilities applied.

geneous team depicts similar training curves. As ω_r increases, the team behavior gets refined by capability over-usage and reconfiguration, which is also why the average episode reward is lower than $\omega_r = 0$. To better evaluate the behaviors, a list of behavior-based metrics in addition to the completion efficiency for both teams and different ω_r is summarized in Table 8.2. When $\omega_r \leq 0.75$, the teams are able to complete tasks at their best levels (low μ_F and high $\rho_{H=10}$). Without compromising the completion efficiency, there is a significant drop in capability usage amount μ_C, idle decision μ_I, and reconfigurations μ_S, as these metrics illustrate refined and effective behavior with the secondary objective being introduced. When $1.0 \leq \omega_r \leq 2.0$, both teams show increased reconfigurations to maintain the same level of capability usage and idling decisions, resulting in

slower completion and lower success rates. When ω_r further increases, the teams tend to avoid reconfigurations as the penalty kicks in, and the performance becomes worse.

Comparing the two team compositions, the homogeneous team has a total extinguishing capability of 10 (1 level higher than the heterogeneous team), but the heterogeneous team can achieve faster completion, lower capability usage, more idling, and fewer reconfigurations. We show the trained behavior of the heterogeneous team in Fig. 8.7 with two cases under uncertainties. The team is able to collaborate and effectively leverage their capabilities and resources. By evaluating the team behavior, we notice that the team does not have enough capabilities to address both fire tasks at the first step, leaving location 1 burning. Also, agent 5 with only 1 level of rescuing capability can potentially be removed if the rescuing tasks at later steps can be assigned to other team members. Such observations on team behavior motivate our co-design problem to find another team composition that can complete the tasks even more efficiently.

8.3.2 Iterative Design and Learning (IDEAL)

Through training with the multi-objective reward, the team gradually shapes the behavior. IDEAL injects heuristic-based composition adjustment throughout the training process, where it seeks to simultaneously improve team composition with heuristics and continue behavior learning with few-shot training. In this section, we present two approaches to achieve the multi-agent team co-design with the IDEAL framework: 1) incrementally removing agents from the initial large pool and 2) iteratively adjusting composition starting from the initial composition.

8.3.2.1 Result: Team Selection—"Given a pool of agents, which ones form the best team?"

We initialize 11 agents with different capability combinations, with extinguishing capability ranging from 0 to 3 and rescuing capability ranging from 0 to 2, and the agent with zero capabilities is disregarded. To allow homogeneity, we duplicate the 11 agents to have a pool of 22 agents at the beginning. In early iterations, we train all agents for 1000 episodes $E_{tr} = 1000$ with secondary objective reward weight $\omega_r = 0.25$ to allow flexibility in capability usage and reconfigurations.

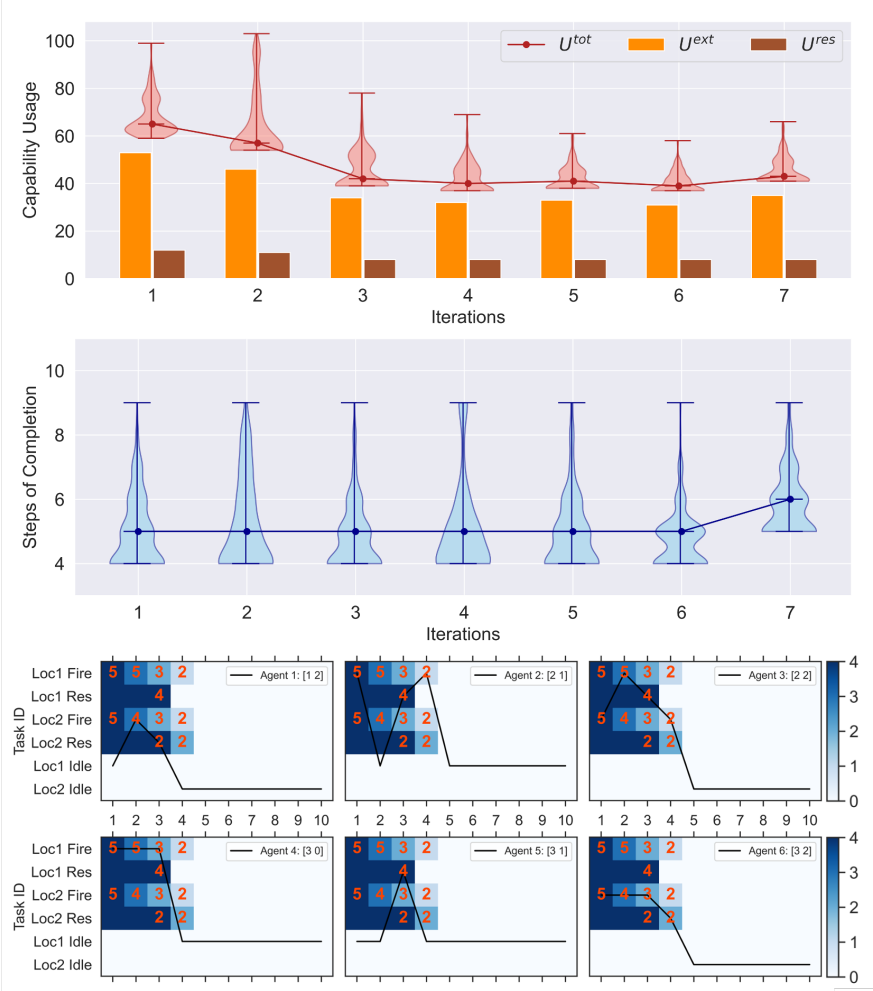


Figure 8.8: Team selection by IDEAL. For each iteration, we show (top) the median of team capability usage for each attribute (U_{ext}, U_{res}) and the distribution of the total capability usage U_{tot}, (middle) the distribution of the number of steps to complete tasks, and (bottom) an episode of the team behavior after iteration 6.

We then gradually increase E_{tr} and ω_r to refine behaviors and increase utility threshold ζ_U select agents. Throughout the process, we use $\omega^{ext}, \omega^{res} = [0.3, 0.7]$ as the utility weights, because less total rescuing capability is needed to complete the tasks (i.e., each rescuing capability unit leads to more improvement than extinguishing capability). The parameters used and the team size for each iteration of the co-design process are shown in Table 8.3.

The team capability usage and completion efficiency for the co-design iterations are shown in Fig. 8.8a,b respectively. The trained behavior of the team after the 6th iteration is shown in Fig. 8.8c. Note that we still keep the extinguishing capability of Agent 5 as it still executes extin-

Table 8.3: Parameters of Team Selection Co-Design Iterations

	Iter1	Iter2	Iter3	Iter4	Iter5	**Iter6✓**	Iter7
N	22	14	12	10	8	6	5
E_{tr}	1k	1k	1k	2k	2k	2k	2k
ω_r	0.25	0.25	0.50	0.50	0.50	0.75	0.75
ζ_U	0.5	0.7	1.0	1.0	1.0	1.0	1.0

guishing tasks in other episodes when the fire level at location 1 does not go down as expected. Also, this agent has utility $\omega^{res} \times c^{res} = 0.7 < \zeta_U = 1.0$ and gets removed before iteration 7. Removing this agent results in a drop in performance (i.e., requiring more steps to complete the tasks). Therefore, we stop the process after 7 iterations and keep the team after the 6th iteration. The final team has 6 agents with the highest completion efficiency and lowest capability usage overall. Although there is an over-usage of extinguishing capability at step 2 for the first fire task, it is reasonable because it accounts for the uncertainties in task dynamics (fire demand not decreasing) and ensures enough capabilities are available for future steps.

8.3.2.2 Result: Team Adjustment—"Given the current team, how to modify and improve?"

We start the co-design processes with the heterogeneous team of 5 agents used in Sec. 8.2.2.1 and demonstrate the design iterations by using two slightly different reward functions on operational requirements: 1) *Process-1 Regular Penalties* $R_D = -\sum_j s_D^j$ and 2) *Process-2 Modified Penalties* $R_D = \min(-4, -\sum_j s_D^j)$, which incentivize faster completion of the operation when only few demand levels remained to complete. For both processes, we train agents with 3k episodes, $\omega_r = 0.75$, utility weights $\omega^{ext}, \omega^{res} = [0.3, 0.7]$ per iteration. To determine the capability need, we impose operational requirement knowledge f_g, and the objectives are to apply sufficient extinguishing capabilities to address all fire demands at every step and to finish rescuing tasks no later than extinguishing tasks. We also inject a resource budget constraint f_b, with which the capability of either attribute for the added agent does not exceed 2.

The statistics of team performance and the heuristics of team adjustment for each co-design iteration are shown in Table 8.4. Throughout the design processes, no agents are found to be less

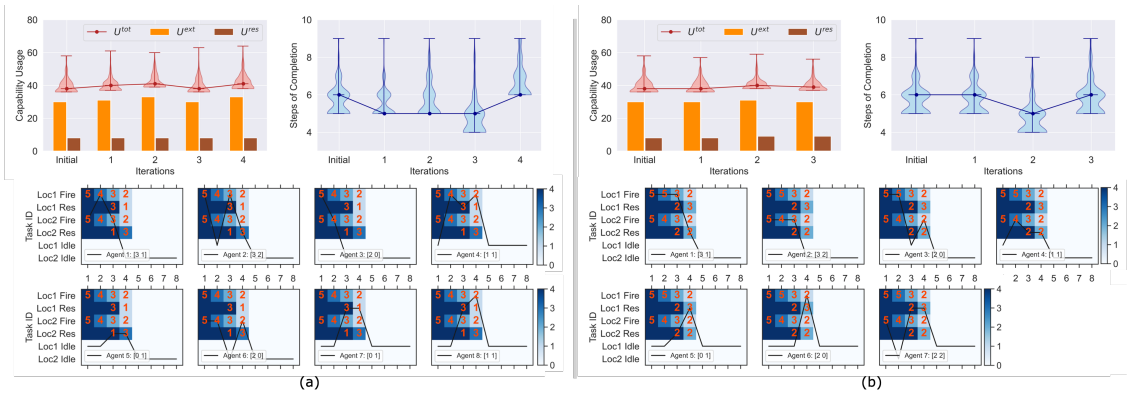


Figure 8.9: Team adjustment by IDEAL starting from the initial 5 heterogeneous agents from Sec. 8.2.2.1. (a) The team receives penalties on total remaining demands $R_D = -\sum_j s_D^j$, and the final team behavior is shown after iteration 3. (b) The team receives modified penalties $R_D = \min(-4, -\sum_j s_D^j)$, and the final team behavior is shown after iteration 4. The top plots show the distributions of capability usage and completion efficiency per iteration for both design processes.

than the utility threshold ζ_U. However, when there is no need to add another agent, we attempt to remove the agent with the lowest utility score to see if the team performance remains similar. Both processes are stopped as removing an agent leads to decreased operational performance. The capability usage and the completion efficiency over the co-design iterations are shown in Fig. 8.9 (top). The teams are able to improve their completion efficiency while maintaining the same level of capability usage. We return the final team for *Process-1* after 3 iterations and for *Process-2* after 2 iterations, and the behaviors of the final teams are shown at the bottom of Fig. 8.9a,b respectively.

8.3.2.3 Discussion

So far, we have demonstrated two co-design approaches by IDEAL and three final teams. We denote the team from the selection approach as Team-1, the team from the adjustment approach Process-1 (regular penalties) as Team-2, and the team from Process-2 (modified penalties) as Team-3. The team compositions and performance statistics are summarized in Table 8.5.

All three teams have similar completion efficiency, but they do have trade-offs in capability usage and team compositions. As we modify the penalties for Team-3 to motivate faster completion, Team-3 tends to apply more-than-needed capabilities to ensure this requirement. Team-1 is

Table 8.4: Team Adjustment Co-Design Iterations

	Size N	**Utility** $\min\{U_i\}$	**Remove** $\zeta_U = 0.5$	**Need** $[c_+^{ext}, c_+^{res}]$	**Add*** c_+
INIT.	5	2.00	Keep	[5.0, 0.0]	[2, 0]
	Regular Penalties				
Iter1	6	1.09	Keep	[0.1, 1.6]	[0, 1]
Iter2	7	1.32	Keep	[1.3, 0.0]	[1, 0]
Iter3✓	8	0.70	Agent [0, 1]	[0.2, 0.3]	None
Iter4	7	0.69	Stop	[5.0, 1.1]	Stop
	Modified Penalties				
Iter1	6	0.68	Keep	[5.0, 2.6]	[2, 2]
Iter2✓	7	0.93	Agent [0, 1]	[0.1, 0.1]	None
Iter3	6	0.80	Stop	[5.0, 1.8]	Stop

* We consider a resource constraint, with which the added agent cannot have more than capability level 2 in either attribute.

Table 8.5: Final Team Compositions and Requirement Satisfaction

	Capability Composition			**Requirements/Episode**	
	Size	**Total**	**Variance**	**Usage**	**Completion**
	N	$[\text{sum}(c^j)]$	$[\sigma^2(c^j)]$	$[\text{med}(U^j)]$	min #.steps
Team-1	6	[14, 8]	[0.6, 0.6]	[31, 8]	4
Team-2	8	[12, 7]	[1.3, 0.4]	[30, 8]	4
Team-3	7	[13, 7]	[1.0, 0.6]	[31, 9]	4

evolved from a large pool of agents. The selection process is also biased toward applying more capabilities as the behavior training has not been finalized in early iterations. However, the selection process can push for a team with a smaller size. Team-2 has the lowest total capabilities overall and is able to apply capabilities without over-usage. This can be achieved because 8 agents with high variance in capabilities provide flexibility when handling heterogeneous tasks. Here, for each co-design process, we only show the final team, but there are other design options in previous iterations to choose from when using IDEAL, as a team with fewer agents might be more cost-effective when taking the manufacturing cost into account.

8.3.3 Vote-Informed Co-Evolution (VICE)

8.3.3.1 Experiment Setup

To validate our proposed evolutionary algorithm VICE, we focus on the team composition evolution part and consider a simplified operation for optimization. We apply VICE with the 3-team format and assume that the computed TS, AS are the outcome of team behavior training. The optimization problem is described here:

- **Team Compositions.** We assume each agent has two attributes v^1, v^2, and each team has three agents ($N = 3$). There exist multiple optimal team compositions

$$\begin{bmatrix} \boldsymbol{v}^1 \\ \boldsymbol{v}^2 \end{bmatrix} = \left[\begin{array}{c|c|c} a & b & c \\ b & c & d \end{array}\right] \text{ or } \left[\begin{array}{c|c|c} b & c & d \\ a & b & c \end{array}\right]$$

  in any column (agent) orders. Each character represents the capability level of the corresponding attribute.

- **Team Score $\boldsymbol{TS}$.** It is analogous to the team performance in the disaster-relief operation. Multiple teams can have high performance, and capability heterogeneity and agent collaboration lead to completion efficiency. The primary metric measures how close the composition is to the optimal one. If only one agent belongs to the optimal, $TS + 1$; if two agents are present, $TS + 3$. The secondary metric counts how many distinct capability levels (characters) are used in the team. For non-optimal compositions, the maximum TS is 9 (i.e., 6 different characters and two agents belong to the optimal composition). If the optimal is found, $TS = 10$.

- **Agent Score $\boldsymbol{AS}$.** It is analogous to the agent utility in the disaster-relief operation, indicating the individual contribution to the team. AS has two metrics. One is how great the character is (e.g. $a = 0, b = 0.1, z = 2.5$), and the other is whether the two characters are in increasing order (e.g., $AS + 1$ if $v^1, v^2 = [a, b]$). As we notice in the disaster-relief

operation, low-capability agents do not necessarily have low utilities and could belong to the final optimal team. These two metrics of TS can model such trade-offs.

8.3.3.2 Results

When applying VICE, we use crossover probability $p_c = 0.45$ and mutation probability $p_m = 0.2$, and the result is compared to the purely random genetic algorithm version without votes (GA Random $p_c = 0, p_m = 0$) as well as Agent-Score Co-Evolution (ASCE) where parents are selected based on agent scores instead of votes. In the 3-team format co-evolution, each team i records the top $K = 2$ team compositions they have seen, and we compute the Team Score Sum (TSS) as the algorithm performance, defined as the total average team score of the top-K compositions $\sum_{i=1}^{3} \mu_i^{TS}$, where $\mu_i^{TS} = \frac{1}{K} \sum_{k=1}^{K} TS_i^k$. The maximum TSS is 30 when the top-2 team compositions are optimal in all 3 teams. TSS is a measure of both solution optimality and diversity, as we encourage VICE to co-evolve multiple teams together. In addition, we track when the first and second solution type is found as there are two types of optimal team compositions. Recall that each solution type can have multiple optimal compositions with different orders of agent.

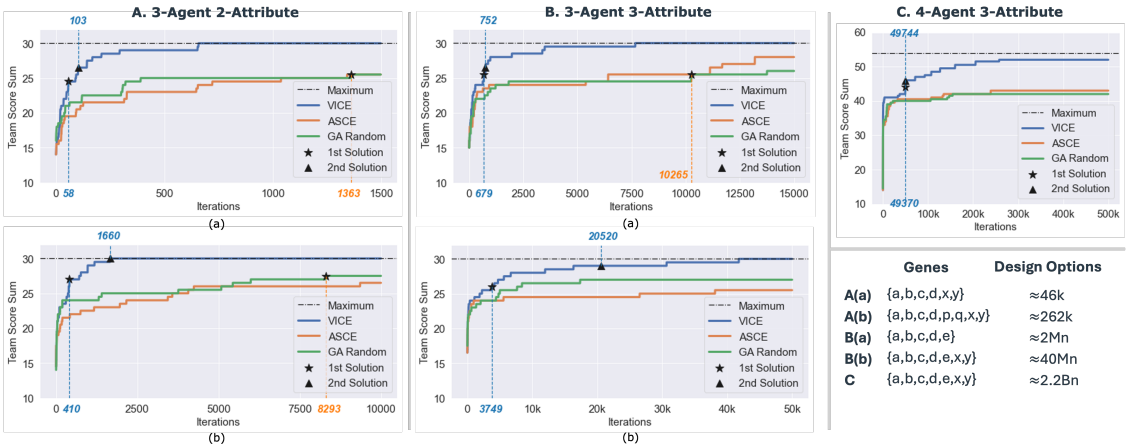


Figure 8.10: Performance of VICE (blue) compared to ASCE (orange) and multi-agent genetic algorithm without votes (green) in different problem complexities. Team Score Sum tracks whether all 3 teams find at least 2 optimal team compositions. The star and triangle symbols indicate when the first and second solution types are found, with the corresponding iteration numbers in color shown near axes.

The results of finding 3-agent 2-attribute compositions with capability level (gene) choices $\{a, b, c, d, x, y\}$ ($\approx$ 46k options) and $\{a, b, c, d, p, q, x, y\}$ ($\approx$ 262k options) are shown in Fig. 8.10 A(a),A(b) respectively. VICE outperforms GA Random in TSS, and VICE can find solution types at least 1 order of magnitude faster than GA Random. Note that the gene choices $\{x, y\}$ resemble the agents with high capabilities and high Agent Score AS, which could make it much harder to find a solution composed of low AS agents. This also explains why ASCE could not find a single solution as it tends to form a team of "highly capable" agents but with a low overall team score. The accumulated votes can help VICE escape this local minimum and inform which agents could lead to the solution. Similar performance is observed when increasing the problem scale to 3-agent 3-attribute with gene choices $\{a, b, c, d, e\}$ ($\approx$ 2Mn options) and $\{a, b, c, d, e, x, y\}$ ($\approx$ 40Mn options) as shown in Fig. 8.10B(a),B(b) respectively. The two solution types are $\left[\begin{array}{c|c|c} a & b & c \\ b & c & d \\ c & d & e \end{array}\right]$ and the one with capabilities in reversed order. Furthermore, in the 4-agent 3-attribute case (Fig. 8.10C) with gene choices $\{a, b, c, d, e, x, y\}$ ($\approx$ 2.2Bn options) and optimal team score $TS = 18$, VICE does not reach the maximum TSS (3×18), meaning not all 3 teams can find 2 optimal team compositions. However, VICE is able to find both solution types in 50k iterations out of 2.2Bn design options.

8.4 Conclusions

The motivation behind the multi-agent team co-design is to fully automate the development cycle of teams composed of multi-attribute agents with strategic planning capabilities to achieve superior operational performance. Instead of finding the teaming strategy given a pre-defined composition, we propose methods to simultaneously co-design team composition and team behavior that belong to different design levels of multi-agent operations. We have demonstrated how the IDEAL co-design framework can select team members from an initial pool of agents and incrementally adjust team compositions to reach superior operational performance in disaster-relief

operations. IDEAL can provide heuristic-based explanations on why certain design changes are made, but the team design outcome would heavily depend on the training and designing parameters and the specific knowledge of operational requirements. In addition, by drawing an analogy between chromosomes and multi-attribute agents, we have shown how the co-evolutionary approach VICE can find multiple operational optimums with a voting mechanism to infer the collective fitness of a team of agents. VICE requires less governance in the design process, allowing agents to evolve through natural selection. However, without proper metrics for individual scores and team scores, the evolutionary approach might find it challenging to solve multi-agent design problems. Given sufficient computational power, leveraging the transition experiences of agents in reinforcement learning to inform the evolution of team compositions could further improve the evolution granularity and facilitate the co-design process.

CHAPTER 9

Dynamic Task Planning for Smart Manufacturing — Applications in Reconfigurable Manufacturing Systems

This chapter presents the applications of the methods discussed in Chapter 3 and Chapter 8 in manufacturing production planning tasks.

9.1 Dynamic Task Planning For Autonomous Reconfigurable Manufacturing Systems by Knowledge-based Multi-Agent Reinforcement Learning

Key enablers of Industry 4.0 and general cyber-physical systems, including artificial intelligence (AI), big data, Internet of Things, and cloud computing, have opened the door to rapid developments in smart manufacturing [190]. Artificial intelligence, autonomous decision-making, and robots have been replacing manual labour, thus massively automating production processes [191]. Meanwhile, humans have started to play supervisory roles, provide domain knowledge, and resolve unanticipated situations, whereas machines are typically assistants in human-machine relationships. However, recent advancements in AI technologies in combination with digital twins [192] have the opportunity to empower machines to become AI-based decision-makers, to collaborate with humans and with each other on complex production tasks, and to adapt under uncertainties, which have not been realized yet and can bring out the untapped potential of machine intelligence.

9.1.1 Introduction

The need for mass individualization in large manufacturing systems requires a transition from conventional centralized planning [193] to decentralized algorithms where systems delegate decision authority to the lowest level [194] and autonomously distribute task planning and production responsibilities to a team of machines [27]. Reconfigurable manufacturing systems (RMSs) are attractive solutions for enabling autonomous factories and machines and offer benefits in adaptability, responsiveness, and scalability when facing unexpected degradation [195] and various product requirements with large quantities [196].

Digital twins of machines allow them to interact in simulation and replicate operational environments in reproducible scenarios as alternatives to exercising in real environments. Multi-agent reinforcement learning (MARL) provides an ideal platform to train a team of machines for collaborative manufacturing tasks. In the literature of manufacturing systems, MARL has been used for planning production tasks and negotiating job assignments by evaluating priorities [197]. In addition, a body of research has utilized reinforcement learning to capture hierarchical task relationships [198] and job shop order scheduling [199]. However, only limited work has focused on forming probabilistic knowledge for learning teaming strategies of modular machines in RMSs.

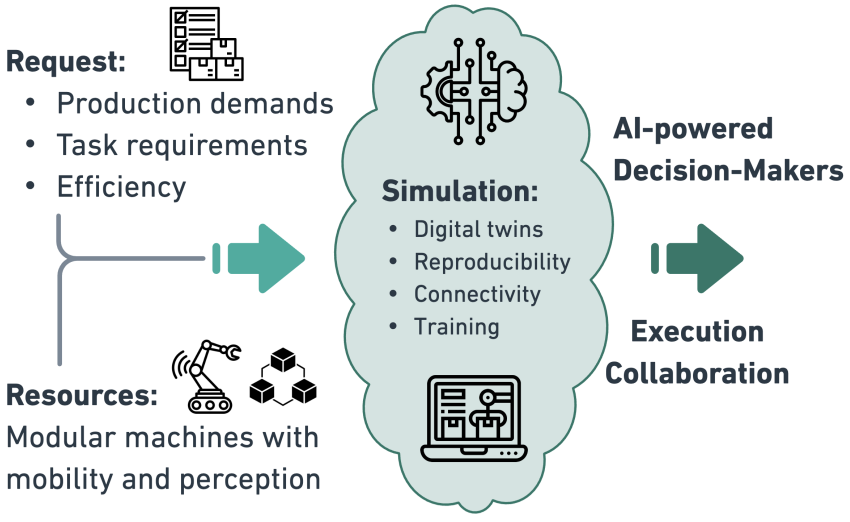


Figure 9.1: Digital twins in reconfiguration manufacturing systems.

Here, we introduce a decentralized MARL approach for future RMSs (Figure 9.1) that have intelligent machines with knowledge-based decision models. These machines are autonomous, can reconfigure modules, reposition themselves, cooperate on tasks, accumulate knowledge through perception and communication, and leverage domain knowledge for task planning and execution. The proposed method is demonstrated in job shop case studies with complex task requirements, and results show effective machine collaboration in several performance measures.

9.1.2 Methods

In this section, we introduce a knowledge-based decision-making framework for a team of reconfigurable modular machines with decentralized decision models using multi-agent deep reinforcement learning. As machines are equipped with their own computing power, decentralized planning is more flexible in response to demand fluctuations and disturbances (e.g., single-machine breakdown), compared to centralized systems.

9.1.2.1 Decision-Making Framework

Machines with autonomous capabilities enable the creation of decentralized manufacturing systems, with enhanced adaptability, because strategic planning authorities can be delegated to each of the machines instead of relying on centralized decisions. In addition to traditional capabilities of sensing, communicating, and reconfiguration, machines require reasoning capabilities to make intelligent decisions and collaborate in complex manufacturing systems. The decision-making process of a team of modular machines can be formulated as an extension of a decentralized partially observable Markov decision process (DecPOMDP) [6] that explicitly models the module-task dynamics (i.e., the effect of module configuration and location on task status) and information exchange. The decision-making framework is defined as a tuple $(M, W, A, S, T, H, O, E, I, R, \gamma)$, where

- $M := \{m_1, \ldots, m_n\}$ is a finite set of n machines,

- $W := \{w_1, \ldots, w_l, \bar{w}_1, \ldots, \bar{w}_l\}$ is a finite set of l workstations and the corresponding standby locations,
- $A := \times_i W, a \in A$, is the set of team actions composed of the decision of each machine m_i operating at a workstation w_j or waiting at a workstation $\bar{w}_j$,
- $S := S_W \times S_M, s \in S$, is a set of all possible manufacturing environment states described by the task state $S_W := \times_j S_W^j$ and machine module configuration state $S_M := \times_j S_M^j$, for each workstation $j \in \{1, \ldots, l\}$,
- $T := \Pr(s' \mid s, a)$ is the probability of transitioning to state s' at the current state s with action a,
- $H := \times_i H_i, h \in H$ is the set of team observations composed of each machine's perceived task status,
- $O := \Pr(h \mid s', a)$ is the observation probability,
- $E : \times_i E_i, e \in E$ is the set of team information composed of the received information from each machine,
- $I := \Pr(e \mid s', h, a)$ is the information probability,
- $R := R(s, a)$ is the reward associated with states and actions,
- γ is the discount factor for the rewards at later steps.

The objective of the autonomous decision process is to find an optimal series of team actions defined as policy π^* for a team of machines that maximizes manufacturing system performance, such as maximizing product quality while minimizing backlogs. Generally, the performance is evaluated by a policy value $V(\pi)$ defined as the expected cumulative reward as follows:

$$V(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s^t, \pi(s^t)) \mid s^0\right], \tag{9.1}$$

where the policy $\pi := S \rightarrow A$ maps the state to the team action, and the optimal policy is $\pi^* = \arg\max V(\pi)$.

9.1.2.2 Knowledge Representation and Probabilistic Reasoning

Using sensing and communication, autonomous machines can acquire information over time and utilize knowledge for reasoning and planning. An effective knowledge representation is important and can prevent the increasing high dimensionality of information gathered over time and space. Thus, instead of storing full histories of action-observation-data, a knowledge state $K = \{b(s), a_c\}$ of a machine can be represented by: 1) a belief state $b(s) = \Pr(s)$, which is a probabilistic distribution over all possible remaining levels of tasks at workstations, and encodes the current system status resulting from past experiences, and 2) the previous configuration a_c needed for reconfiguration tracking. To avoid accumulating large data in a time-dependent state, a knowledge update mechanism is implemented to update the two components of the knowledge at each time step. Specifically, the previous configuration tracks the module usage and workstation location of the last decision a, and is updated by $a_c' = f(a)$, where the prime symbol indicates the next time step. The belief state update follows the Bayes' rule as shown in Eq. 9.2 using the current belief state b, action a, perceived observation h, received information e, and knowledge of the corresponding probabilities (i.e., T, O, I), namely:

$$\begin{aligned} b'(s') &= \Pr(s' \mid e, h, a, b) \\ &\propto \Pr(e \mid s', h, a) \Pr(h \mid s', a) \Pr(s' \mid a, b) \\ &\propto I(e, s', h, a) O(h, s', a) \sum_s T(s', s, a) b(s). \end{aligned} \tag{9.2}$$

9.1.2.3 Decentralization and Problem Transformation

Instead of relying on the operating environment for reward generation or a centralized decision-making system in MARL, each machine can find the optimal policy given its knowledge state and a self-rewarding mechanism (i.e., each machine rewards itself based on the knowledge state). The

index i of each machine in a team is omitted for the rest of the section because the problem is completely decentralized. This new decision-making framework is referred to as knowledge-based DecPOMDP (KDP). The objective of this problem is then to find the policy for each machine that maximizes the Q-value $Q^{\pi}(K, a)$, defined as the expected cumulative reward at any given knowledge state K by taking an action a. The Q-value follows Bellman Optimality, and is given by:

$$Q^{\pi}(K, a) = \mathbb{E}_{K'}\left[R(K, a) + \gamma \max_{a'} Q^{\pi}(K', a') \mid K, a\right]. \tag{9.3}$$

Designing adequate reward functions for machines is crucial for forming expected teaming behaviors and satisfying manufacturing objectives. In RMSs, modular machines can reconfigure modules, reposition themselves to different workstations, and stay idle. Their teaming behaviors might lead to module over-usage, long ramp-up periods, and production delays. Thus, we construct the reward function $R(K, a) = R(b, a_c, a)$ based on three objectives: faster operation completion, fewer reconfigurations in terms of modules and workstations, and a lower module usage as:

$$R(K, a) = R_{op}(b) + \omega_1 g_{conf}(a_c, a) + \omega_2 g_{mod}(b, a), \tag{9.4}$$

where $R_{op}(b)$ generates rewards when all tasks are completed in the belief state, $g_{conf}(a_c, a)$ is the reconfiguration cost when a machine is reconfiguring from a_c to a, and $g_{mod}(b, a)$ penalizes unnecessary module usage (e.g., operating at a workstation with no modules needed). ω_1, ω_2 are tuning parameters to scale down penalties as operation completion is the primary objective.

9.1.2.4 Training by Deep Q-Learning

By interacting with each other in a simulated environment, machines can learn from experience through reinforcement learning. Each machine is equipped with a deep Q-network $Q(K, a; \theta)$ as the decision model to approximate the Q-value so that $Q(K, a; \theta) \approx Q^{\pi}(K, a)$, and the optimal policy can be retrieved by $\pi^*(K) = \arg\max Q(K, a; \theta)$. From Eq. 9.3, the Q-value converges to

a target Q-value

$$y(K, a, K'; \hat{\theta}) = R(K, a) + \gamma \max Q(K', a'; \hat{\theta}), \tag{9.5}$$

where $Q(K', a'; \hat{\theta})$ is a target Q-network with parameters $\hat{\theta}$ updated by θ less frequently during training for stability.

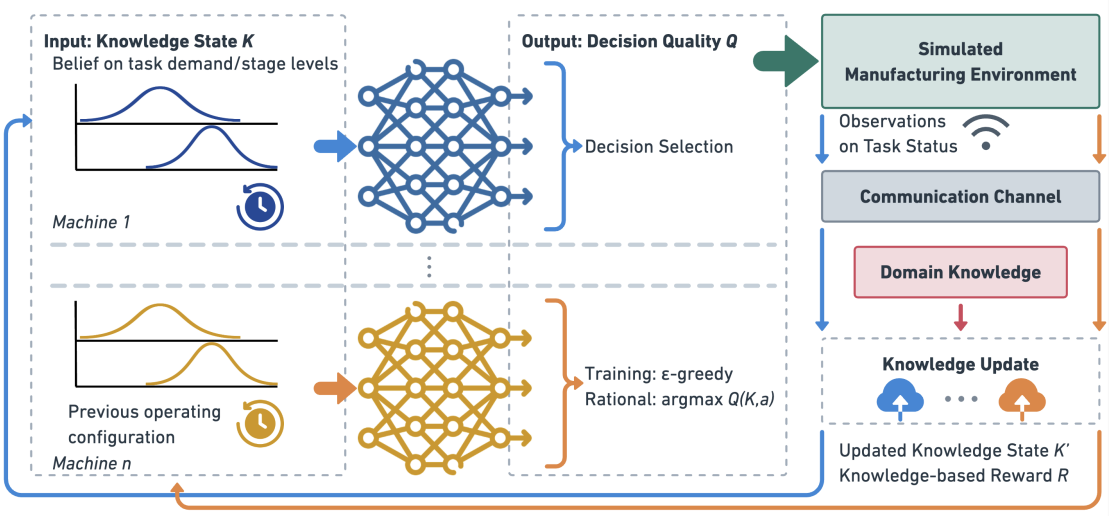


Figure 9.2: Decentralized deep Q-learning for the KDP framework.

By interacting with the environment through exploration and exploitation (Figure 9.2), each machine selects a decaying ϵ-greedy action a at the current knowledge state K, receives observation and information from its sensors and possibly from other machines, updates the knowledge K', computes the reward R, and stores the experience training data $x = (K, a, K', R)$ into a replay memory D. The training for the Q-network minimizes the loss $L(\theta)$ between the Q-value $Q(K, a; \theta)$ and the target Q-value y using a batch of random samples $X = \{x\}$ from D, namely:

$$L(\theta) = \mathbb{E}_{X \sim D}\left[y(K, a, K'; \hat{\theta}) - Q(K, a; \theta)\right]. \tag{9.6}$$

9.1.3 Results

To demonstrate the application of the proposed method, two production job shop case studies are adapted from [1] based on example parts used in the RMS literature (Figure 9.3). The part

instances consist of features (holes, surfaces, etc.) that require different manufacturing tasks as well as different types and combinations of machine modules to execute those tasks.

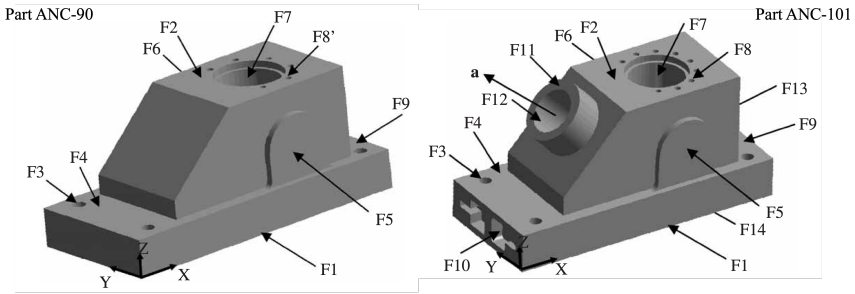


Figure 9.3: Example parts with different features, adapted from [1].

9.1.3.1 Case 1: Collaboration of Specialized Machines

We consider a production operation with two specialized modular machines and four task workstations (Figure 9.4). Each machine is equipped with one module of each type, and each workstation is responsible for one feature of the part. When a production request is received, the operation could be formulated by capturing the detailed dynamics for each time step (every second) for each product demand quantity. However, demands are dynamic, and the rapid change requires frequent re-planning. Thus, we represent demands as the quantity level of products remaining to be manufactured, which captures the task dynamics and relationships equally well. In the case study explored herein, a product with demand level 2 requires four different features to be manufactured at four workstations. Workstations CoopA and CoopB require both module types and the cooperation of two machines (i.e., machines configured with those modules). Arrows in Figure 9.4a indicate product flow and task sequentiality requirement (e.g., machines can start working at CoopB only after the product demands at SpecA and SpecB are both accomplished, but machines can work at SpecA and SpecB simultaneously). To account for uncertainties, we deliberately add a high delay rate of 50% for the SpecB task, i.e., with appropriate modules, the probability for the demand level of the SpecB task to decrease by 1 level is 0.5. For simplicity, the other tasks have zero delay rates.

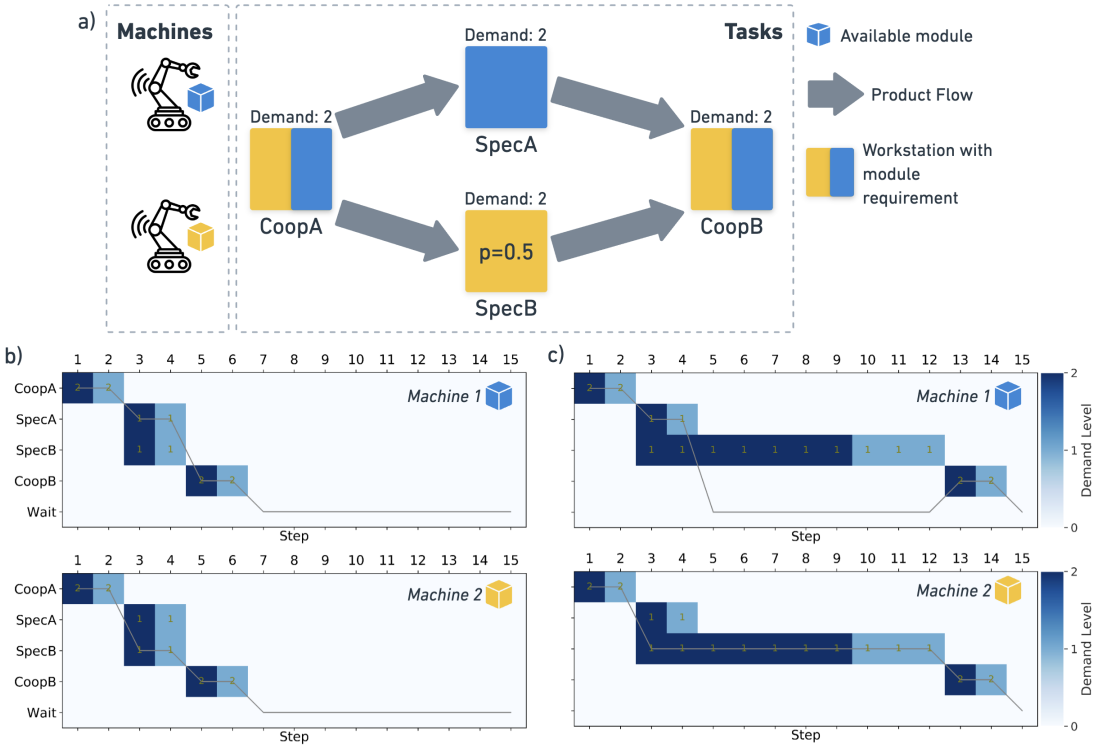


Figure 9.4: a) A production scenario for two specialized machines; Task plan visualizations in b) an optimistic case and c) a pessimistic case.

After training, the behaviors observed in two episodes are shown in Figure 9.4b and 9.4c. At each time step, colored cells represent the demand levels of available tasks, lines indicate machine decisions over time, and numbers show the number of machines working at each task. In an optimistic episode (Figure 9.4b), the SpecB task demand decreases without delays as Machine 2 is working at SpecB, and two machines learn to collaborate and accomplish all tasks efficiently. In a pessimistic episode (Figure 9.4c), while the high delay rate of SpecB is preventing Machine 2 from finishing the job, Machine 1 learns to stay idle and waits until the CoopB task becomes available.

9.1.3.2 Case 2: Collaboration of Reconfigurable Machines

To further demonstrate the adaptability and performance of the proposed KDP framework, we model a more complex job shop operation (Figure 9.5), considering four heterogeneous modular machines and three production lines with multi-stage tasks at five workstations. In task execution, machines choose an active module type, and other modules remain inactive. There are three types

of tasks with different task-module dynamics, defined as follows:

- Type-1 tasks at Station A and C: Stage 1 requires at least 3 modules of Type-1; Stage 2 requires the product from Stage 1 and at least 1 module of Type-1; demands go down by 1 level for each step.
- Type-2 tasks at Station B and D: Stage 2 requires products from Stage 1 of both stations; demands go down by the number of Type-2 modules for each step.
- Type 3 tasks at Station E: the task requires at least 1 module of each type; demands go down by 1 level for each step.

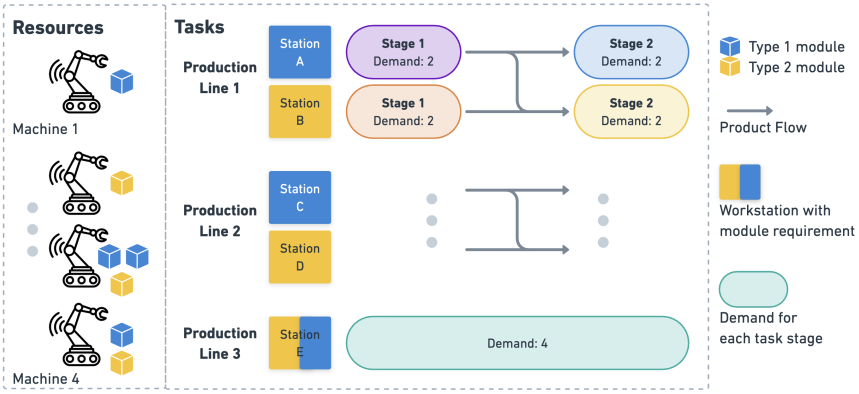


Figure 9.5: A production scenario with 4 reconfigurable machines.

To complete Type-1 and Type-2 tasks, products of demand level 2 need to be manufactured for each of the two stages. To complete Type-3 task, products of demand level 4 need to be manufactured. The process is accomplished when all tasks reach demand level 0. A 10% delay probability for Stage 1 of Type-1 tasks and Task-3 is added to reflect that they require a large module amount or both module types. In this study, machines can only observe the demands at their operating workstation, and the observations are accurate (i.e., machines have robust sensing). The communication between machines is assumed to be all-to-all, and only the observed task demand levels are being communicated with no other information on past decisions or intentions.

To investigate the effects of reward shaping on behavior formation and operation performance, we conduct a parametric study on the secondary objectives of the operation (i.e., the amount of

penalty on module reconfiguration and module over usage), while the primary objective (fast completion) ensures the efficiency of operation. The MARL training process of four different levels of penalty (No/Low/Medium/High) is shown in Figure 9.6a, where shades indicate the standard deviation of the training reward. As expected with the higher penalty, the team of machines spends more time exploring the optimal strategy and refining their behavior and eventually converges to a similar reward value. Because no penalty is applied to the secondary objectives, the reward is slightly higher than the ones obtained with penalties.

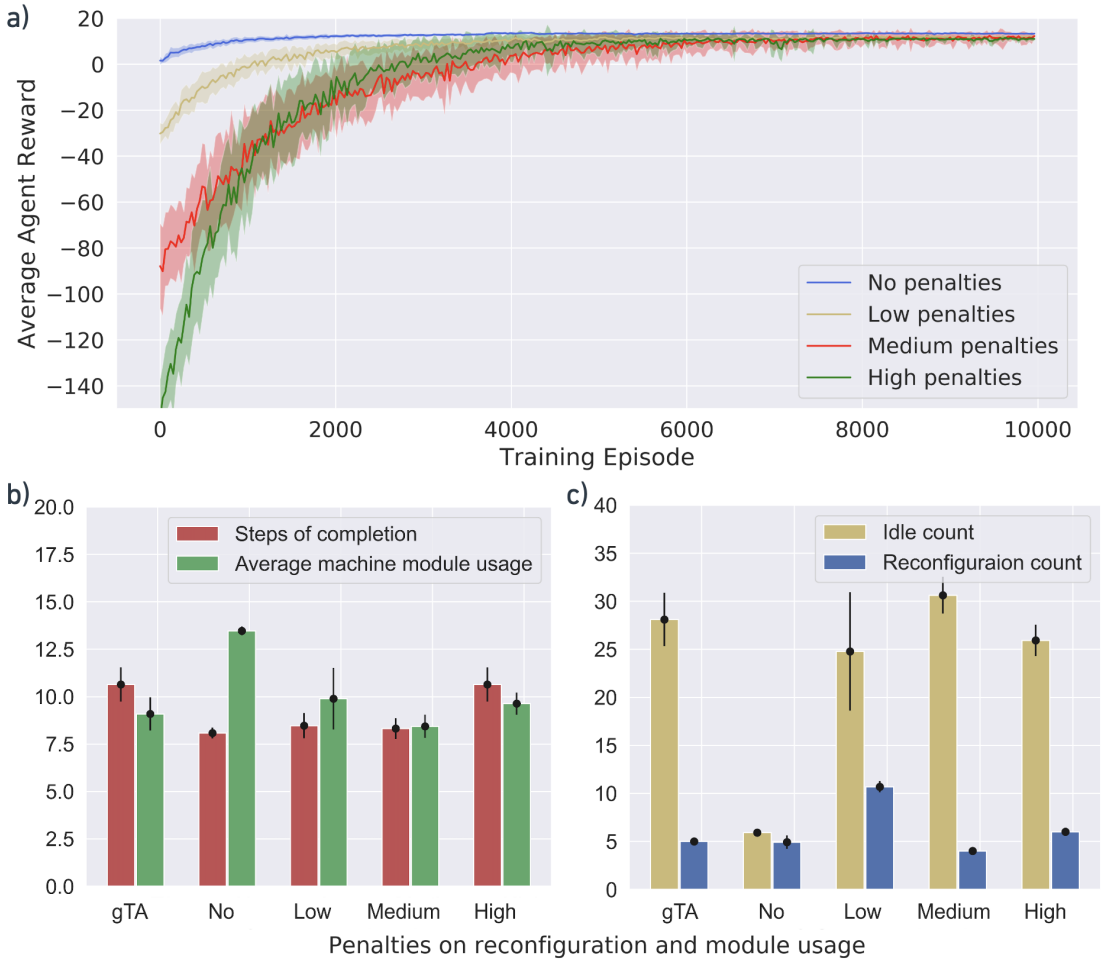


Figure 9.6: Training (a) and performance measures (b)(c) for different levels of penalties on reconfiguration and module usage.

For additional validation, we compare our method to a greedy auction-based task allocation algorithm (gTA) [200], where machines bid for the task that brings more utilities. We consider

four performance metrics: 1) number of steps for completing all tasks (red), 2) average module usage for each machine within 15 steps (green), 3) frequency/count of the Wait action (yellow), and 4) frequency/count of module reconfiguration and production line switches (blue). These metrics are measured over 200-episode realizations for each penalty level. The average and the standard deviation of the metrics are shown in Figure 9.6b and 9.6c. Results show that, compared to gTA, the proposed method can achieve faster task completion with less module usage and fewer reconfigurations. With increasing penalties on secondary objectives, the team operates with less module usage, fewer reconfigurations, and more idling behavior, all these while maintaining the completion efficiency (≈8.2 steps). When the penalty is too high, machines choose to not reconfigure themselves, resulting in slower demand satisfaction, higher module usage and less idling.

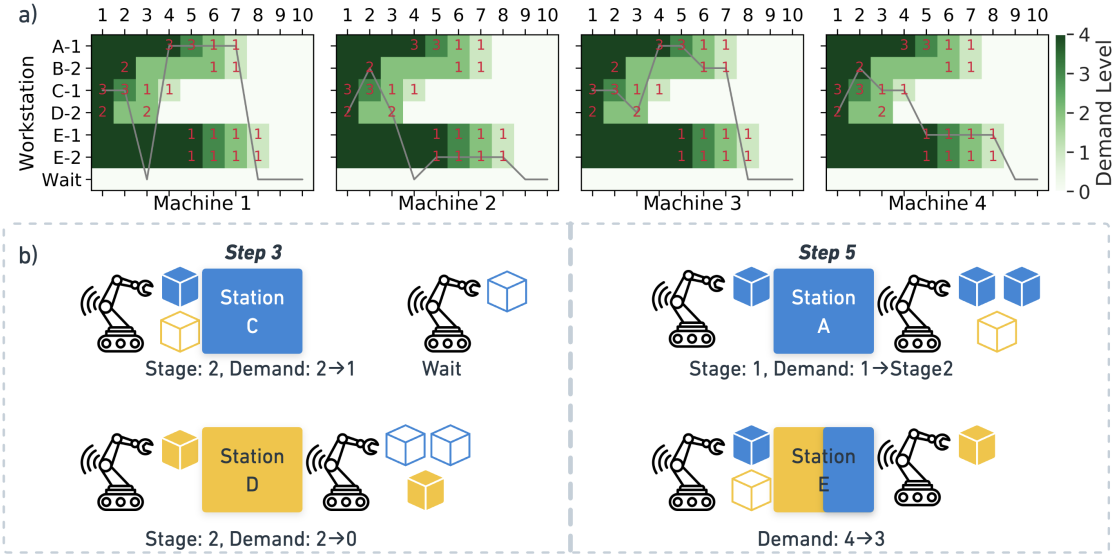


Figure 9.7: Strategy visualizations for 4 reconfigurable machines, with a) team decisions over 10 steps and b) strategy snapshots at step 3 and 5.

Among all penalty levels, the team trained with Medium penalty is selected to visualize the trained strategies (Figure 9.7). For clarity, we consider 4 demand levels (2 demands for each of the 2 stages) for each workstation task. Tasks at stations A, B, C, D are module-specific (i.e., they either require a Type-1 module or a Type-2 module). Station E task requires both module types, and thus two configurations (E-1, E-2) are shown to indicate the corresponding module type applied.

Colored cells represent the remaining demand levels of the tasks, numbers indicate the module usage at each task/configuration, and lines track the decisions of machines. Results show that machines learn to collaborate on accomplishing tasks, reconfigure modules, and autonomously relocate to another production line or workstation when necessary. The modules are effectively allocated without over-usage. For example, in step 3, Machine 2 and Machine 3 cooperate on the Type-2 task at Station D and reduce the demand to 0, Machine 4 activates its Type-1 module and completes the task at Station C, and Machine 1 is waiting since no more Type-1 module is needed. At step 5, four machines pair up to operate at two workstations.

9.1.4 Conclusions

A decentralized knowledge-based decision-making process (KDP) is proposed for efficient and collaborative teaming of RMSs composed of autonomous modular machines in production tasks. Such RMSs allow machines to gather information, maintain probabilistic knowledge, and learn teaming strategies through multi-agent deep reinforcement learning. The proposed method also includes a multi-objective reward function to regulate machine reconfigurations and prevent module over-usage without harming the effectiveness of demand satisfaction. The responsiveness of autonomous machines was demonstrated in two studies. The first study reveals the efficiency of the approach in an optimistic case without production delays and highlights the adaptivity in a pessimistic case with probabilistic production delays. The second study shows the training effectiveness with the carefully designed reward function with quantified teaming behaviors into four performance metrics. To ensure the deployment of such machines, studies on system scalability, decision-making under discrepancies in knowledge and facts, and machine movement feasibility would require further investigation. Potential future work includes exploring teaming of machines with human workers, exploring efficient communication strategies to transfer only the necessary information, and developing methods to efficiently design the team of autonomous modular machines.

9.2 Intelligent Strategy for Learning and Teaming in Manufacturing Systems with Collaborative Autonomous Modular Machines

To achieve smart manufacturing, decentralization, and reconfigurability are essential to providing flexibility and scalability. Here, an intelligent anarchic learning paradigm is proposed to empower modular mobile machines with knowledge-based AI models to collaborate with each other and with human operators. Instead of separating the problems of strategy learning and modular design, an iterative design-while-learn framework is introduced to enhance efficiency and adaptability in reconfigurable manufacturing systems using decentralized deep reinforcement learning. A shop floor operation is simulated to illustrate the effectiveness of the proposed methods in response to production uncertainties and dynamically changing demands.

9.2.1 Introduction

The success of smart manufacturing systems (SMSs) has been enabled by advanced computing technologies (e.g., cloud computing), increased data availability (e.g., Internet-of-Things), and human-level artificial intelligence (e.g., deep learning) [190]. To achieve fully autonomous and highly adaptable manufacturing, SMSs require various advanced capabilities including intelligence, responsiveness, flexibility, and scalability [201]. Traditional manufacturing systems rely on specialized machines for efficient automation and involve human knowledge and supervision to address unanticipated disruptions. Recent progress in Industry 4.0 technologies has accelerated distributed planning and control as well as reconfigurable manufacturing systems to address an increased need for mass individualization and delivery of highly customized products of high quality with limited resources.

By taking advantage of cyber-physical systems [202], machines are transitioning from centralized control to independent autonomous decision-makers that can store, perceive, exchange information, and even synchronize knowledge with human expertise [203]. Decentralization, motivated by scalability and resilience, leads to anarchic manufacturing systems (AMSs), an individualized

market-based distributed planning that shifts autonomy to the lowest level entities [204]. Because of the self-organizing nature of AMSs, logical and explainable reasoning become important elements, and further methods to achieve deep collaboration among machines and human operators are needed.

AMSs empower reconfigurable planning and modular design [205] in both production networks [196] and production lines [206] that effectively tackle the stochasticity in demands and the uncertainty in manufacturing task requirements [207]. Limited research has considered operator modularity in reconfigurable manufacturing systems (RMSs), where machines are equipped with different functional modules for various collaborative tasks. The modularity, augmented by high-level adaptability and autonomy, are key enablers for human-centric manufacturing in Industry 5.0.

AMSs and RMSs are emerging as vital components of smart factories. Another indispensable technology driving innovations in SMSs is machine intelligence [208]. Deep learning techniques such as deep reinforcement learning have been developed to understand the underlying dynamics of unstructured data and to reason with complex features. Such methods have been applied in manufacturing diagnosis and task scheduling problems [209, 210].

This section presents a novel intelligent anarchic learning (I-AL) paradigm, synthesizing various aspects of SMSs including decentralization for scalability, knowledge-based AI for decision responsiveness, and machine reconfiguration for adaptability [1]. The proposed I-AL seeks the ultimate autonomy and efficiency in manufacturing systems by preparing reconfigurable machines in simulated or real environments to collaborate with each other and with humans on tasks with complex requirements [211]. In I-AL, a team of autonomous agents (e.g., machines) maintains individual knowledge, adaptively reconfigures members, and makes strategic decisions to achieve optimal, adaptive manufacturing operations.

The ability of the SMS to learn strategies with fixed resources (i.e., for a given inventory of machines and modules) is extended to solve by intelligent team design customization. Specifically, a design-while-learn (DwL) framework is proposed to combine learning and production objectives

into an iterative process. DwL finds effective teaming strategies, utilizes these learned strategies to infer the gap between the current performance and the desired outcome, and then adjusts team compositions considering resource constraints. With the aid of iterations and domain knowledge, DwL also has the important potential of adjusting strategies to evolving situations and unexpected events/demands.

The proposed I-AL and DwL are demonstrated in a shop floor operation with complex task demand dynamics. In particular, the following three research questions are addressed:

1. How to incorporate robust knowledge representation and maintenance under uncertainties in SMS operations?

2. How to make intelligent decisions while considering the trade-offs in reconfiguration and task efficiency?

3. What are the influential factors and heuristics for designing optimal team compositions adaptive to fluctuating demands?

Advanced manufacturing systems can be viewed as a team composed of machines and humans. Among these systems, centralized or hierarchical manufacturing systems rely on human expertise while machines are subordinate members of the team. Humans have full and exclusive authority over decisions. With computer integration, machines in heterarchical manufacturing acquire more decision authority and are able to coordinate according to specialization, while the team roles are still predetermined. AMSs delegate decision authority to all members of a team, where humans and machines have equal responsibilities in strategic planning and control. Such an architecture relaxes human supervision and cognitive workload, ensures scalability with agent-based modeling, but requires an abundance of machine intelligence and adaptability.

The I-AL introduced here prepares machines for responsive collaboration on manufacturing tasks, with key enablers shown in Fig. 9.8. Inheriting the anarchic architecture, machines are integrated with perception capability to observe task status and share information, although information may be imperfect due to limited sensors and communication. By taking advantage of

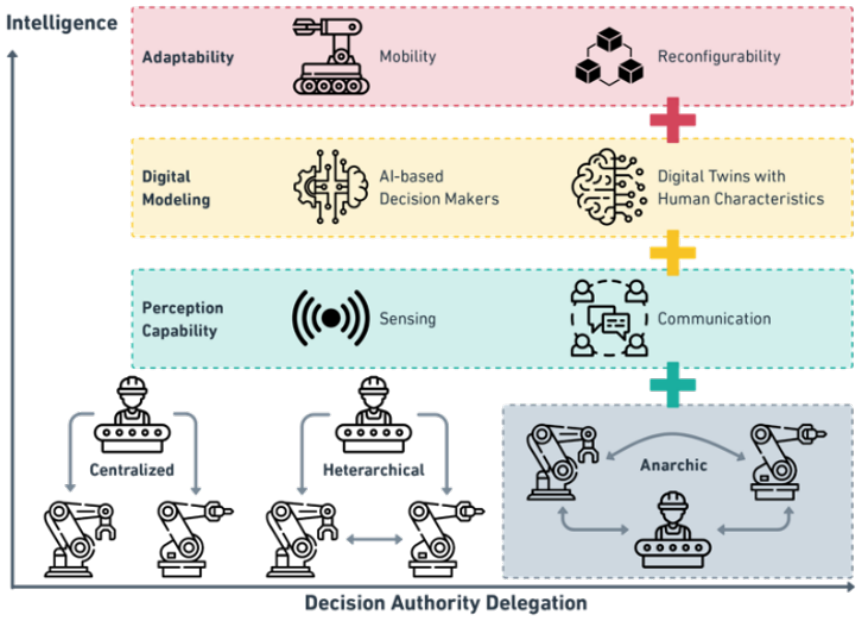


Figure 9.8: Components of Intelligent-Anarchic Learning paradigm.

advanced models, I-AL creates digital twins of team members, for clarity/simplicity referred to as agents [212]. Agents are equipped with individualized AI-based decision models with their own preferences, and they are trained by deep reinforcement learning to capture complex features such as temporal dynamics and uncertainties. To collaborate with human operators in real tasks, it is necessary for agents to experience human behaviors during training. Hence, digital human agents are modeled to incorporate human characteristics (e.g., creativity, varying satisfaction, adaptation to surprises) for realistic interaction [27]. To achieve superior effectiveness under the challenge of rapidly evolving demands and product variety, agents are also autonomous and mobile so they can relocate themselves on the factory floor [213] and reconfigure themselves through module substitution. Trade-offs in task completion and reconfiguration are also embedded into the decision-making process.

To demonstrate the flexibility and responsiveness of the proposed I-AL, a decision step of 3 trained agents in a 4-station factory shop floor operation is shown in Fig. 9.9. The station blocks denote 4 tasks with different module requirements (red, blue, or both), and filled blocks indicate higher product demands. The grey arrows represent the sequentiality of the tasks. Agents

make decisions based on the task requirements and demand status, and they can collaborate or individually work at any station.

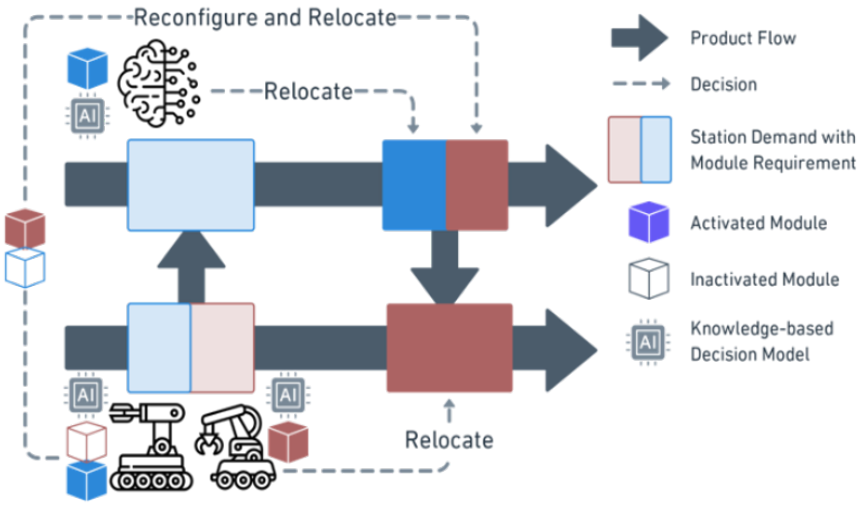


Figure 9.9: Flexible and collaborative decision-making in I-AL.

9.2.2 Methods

This section describes the training pipeline of I-AL and extends the strategic planning problem to the system-level design. An iterative framework, consisting of strategy learning and team composition design, is introduced.

9.2.2.1 Problem Formulation

During training, agents are allowed to interact with a simulated environment using a reinforcement learning paradigm that accounts for uncertainties due to information completeness, human-machine interaction features, production task delays, and sensor accuracy. In I-AL, the multi-agent teaming problem can be formulated by a Module-based Decentralized Partially Observable Markov Decision Process (ModDec-POMDP), defined as a tuple $(S, A, T, M, I, R, \gamma)$ described in Table 9.1. For readiness, index i is omitted in single-agent representations in the rest of the text. The objective of the problem is to find sequences of decisions that maximize the expected cumulative rewards of the agents.

Table 9.1: ModDec-POMDP formulation

Symbol	Description
$s \in S$	State described by s_d: operation task demand/stage and s_c: requirement-specific module configuration, $s = \{s_d, s_c\}$
i	Agent index
$a \in A$	Team action, $a = \{a_i\}$
T	Probability of reaching the next state s', $\Pr(s' \mid s, a)$
$m \in M$	Team information, $m = \{m_i\}$
I	Probability of receiving information m, $\Pr(m \mid s', a)$
R	Reward depending on state-action pairs, $f_R(s, a)$
γ	Discount factor to account for future reward

9.2.2.2 Knowledge Representation and Update

With the presence of uncertainties in operations, an agent makes decisions based on its accumulated knowledge since the actual system-level state is not directly accessible. As shown in Fig. 9.10a, an agent's knowledge state $K = \{b(s), a_o\}$ consists of a belief state $b(s)$ and previous configuration/decision a_o, where $b(s)$ is the inferred probabilistic distribution over operation demands that summarizes the agent's action-information histories. When updating the knowledge state $K' = \{b'(s'), a\}$ for the next time step, the current action a becomes the previous action, and the belief state is updated by using the probability I of receiving information and the probability T of reaching the next state, i.e.

$$b'(s') = Pr(s'|m, a, b) \propto I(s, a, m) \sum_{s} T(s, a, s')b(s). \tag{9.7}$$

9.2.2.3 Reward Shaping

A well-designed reward function is crucial because it can lead to a fast and reliable learning process. This is especially important in the multi-objective I-AL paradigm, where an agent might be trapped in local minima (e.g., unwilling to reconfigure, or is satisfied with partial task completion). Thus, the reward function given the knowledge state and action $f_R(K, a) = f_R(b(s), a_0, a)$ is defined according to two objectives: i) fast operation completion as the primary objective, and

ii) reduced module usage and fewer reconfigurations. Therefore, the reward can be expressed as

$$f_R(K, a) = R_{op}(b) - U_{mod}(a) - C_{config}(a_0, a), \tag{9.8}$$

where R_{op} is the comparably high reward for completing all tasks in the operation, $-U_{mod}$ penalizes unnecessary module usage given task requirements, and $-C_{config}$ penalizes reconfiguration due to relocating and module switching. Note that the reward can be individualized to model various types of agent behaviors.

9.2.2.4 Decision-Making by Decentralized Deep Q-Learning

Under Bellman Optimality used in deep reinforcement learning, the action quality at a knowledge state $Q(K, a)$, simply referred to as the Q-value, is given by

$$Q(K, a) = \mathbb{E}_{K'}\left[f_R + \gamma \max_{a'} Q(K', a') \mid K, a\right]. \tag{9.9}$$

For each agent, a deep Q-network $Q(K, a; \theta)$ parameterized by θ is constructed to approximate the Q-value. The Q-value converges to the target Q-value $y = f_R + \gamma \max Q(K', a'; \phi)$, where parameters ϕ of the target Q-network $Q(K', a'; \phi)$ are updated with θ less frequently. The optimal policy π^* represents the solution of the optimization $\pi^* = \arg\max_a Q(K, a; \theta)$. Through interactions in simulation as shown in Fig. 9.10a, each agent selects an action a given the current knowledge K, receives information to update knowledge K', computes a reward f_R, and stores the experience transition data $x = \{K, a, K', f_R\}$ into its replay memory D. The training process minimizes the loss $L(\theta)$ between the Q-value $Q(K, a; \theta)$ and the target Q-value $y(K', a'; \phi)$ using a batch of transitions $X = \{x\}$ randomly sampled from the replay memory D:

$$L(\theta) = \mathbb{E}_{X \sim D}\left[y(K', a'; \phi) - Q(K, a; \theta)\right]. \tag{9.10}$$

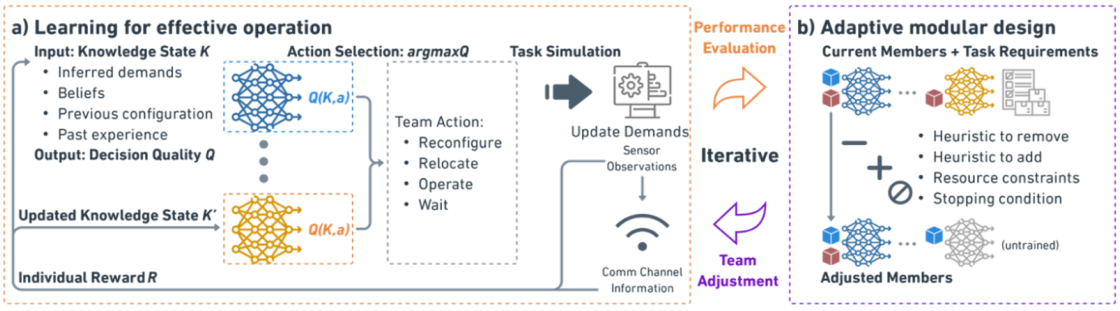


Figure 9.10: Design-while-Learn framework has an iterative process of (a) I-AL using deep reinforcement learning and (b) heuristic-based modular design.

9.2.2.5 Adaptability and Performance by Design-while-Learn (DwL)

The proposed I-AL generates responsive decision-making solutions for the team of machines and humans while flexibility is embedded in the decision level only because agent modules are predefined and task requirements are constant in DwL. The proposed DwL framework (shown in Fig. 9.10) solves the strategy learning and the modular design problems both at once. In particular, DwL iteratively adjusts team members given available resources and desired task performance with minimal retraining.

During iterations, after the first training step, the operation is likely not optimal in terms of module usage and efficiency. Moreover, task requirements might change during iterations. Consequently, some team members may not be needed anymore, or sometimes another machine with a certain module composition can perform better without excessive module usage. The proposed adaptive modular design (AMD) shown in Fig. 9.10b determines: 1) whether to remove an existing member with low usage, 2) whether to add a member with the most needed module composition under resource constraints, and 3) whether to terminate the process. The heuristics regarding these 3 decisions are summarized in Table 9.2.

9.2.3 Experiment and Results

To demonstrate the effectiveness of the proposed I-Al and DwL, a manufacturing operation with two production lines and four stations is simulated (Fig. 9.11), and synthetic data is generated

Table 9.2: Adaptive modular design (AMD) heuristics

Design Step	Heuristic
Remove	• Quantify agent utility using module usage • Remove agents with utility below a threshold or the agent with the least utility
Add	• Identify the most needed module according to task requirements and state-action histories • Apply resource constraints
Terminate	• No change in team members • Decrease in operation performance

for training agents. For ease of validation, the production lines are considered to have the same task requirements. Each production line has two stations, and the tasks at stations are model-specific. Task stations A and C are defined as Type I tasks requiring Module I to be handled, and task stations B and D are Type II tasks requiring Module II to be handled. To incorporate collaboration and task inter-dependency, two types of task dynamics are explored, as follows. For Type I, more modules are needed to reduce the demand at Stage 1 compared to Stage 2. For Type II, stage demands decrease proportionally to the number of modules, but Stage 2 requires products from Stage 1 of both stations. The possible delay in production is modeled as uncertainty in task transition probabilities. Agents need to collaborate on the same task or work synergistically on different tasks to meet all the demands. The objective of the developed framework is to train a team of agents that effectively satisfies the task demands without module over-usage and with minimum reconfigurations. Module requirements, demand level splitting, and product flow can be easily customized to model any complex scenario.

First, a team of 4 reconfigurable agents is selected and trained (Fig. 9.12a) to satisfy task requirements in a shop floor operation. To illustrate the importance of reward shaping in the

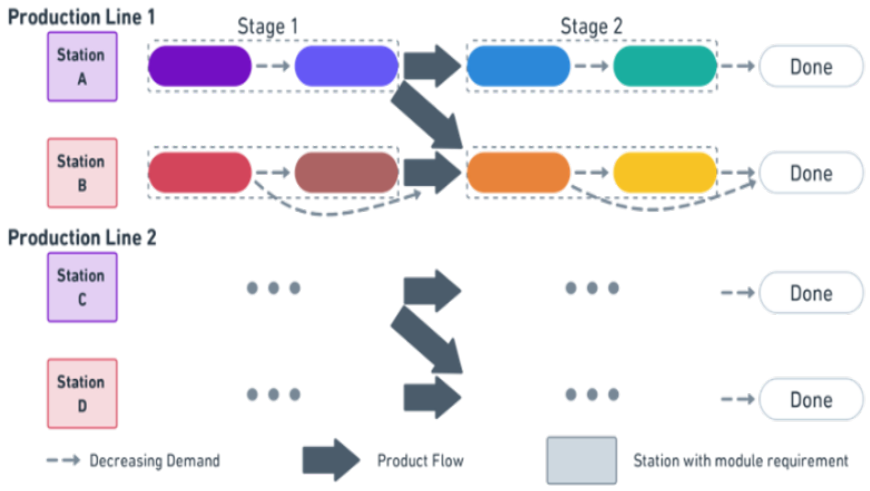


Figure 9.11: Task requirements in a shop floor operation.

outcome of the reinforcement learning process, multiple objectives of completion, module usage, and reconfiguration are considered as described by Eq. 9.8 and compared with the results from a reward that has a single objective for completion. Multi-objective reward (MOR) spends more time exploring an optimal solution with minimum module usage and reconfiguration, and it converges to a slightly lower value compared to a single-objective reward (SOR) approach. Although apparently small, the difference in objectives completely changes the teaming behavior. An example episode is shown in Fig. 9.12b. It takes 6 steps to complete the production tasks for both teams (trained by MOR and SOR separately), but the team trained by MOR operates better because agents do not unnecessarily relocate or reconfigure, and they change to standby (wait action) when not needed.

Next, we explore two strategies to design the team while simultaneously training the agents in the team. Fig. 9.13 shows the first design strategy where a final team of 4 heterogeneous agents is selected from a pool of 16 agents with module amounts ranging from 0 to 2 after 7 iterations. The team design is performed by starting from an initial pool of agents and iteratively removing agents with minimal contribution to the production line, i.e. agents with at most 1 task assignment. This strategy reduces the module usage to the minimum number without hurting performance. The process stops when the performance is measured by the completion step and the cumulative reward plummets.

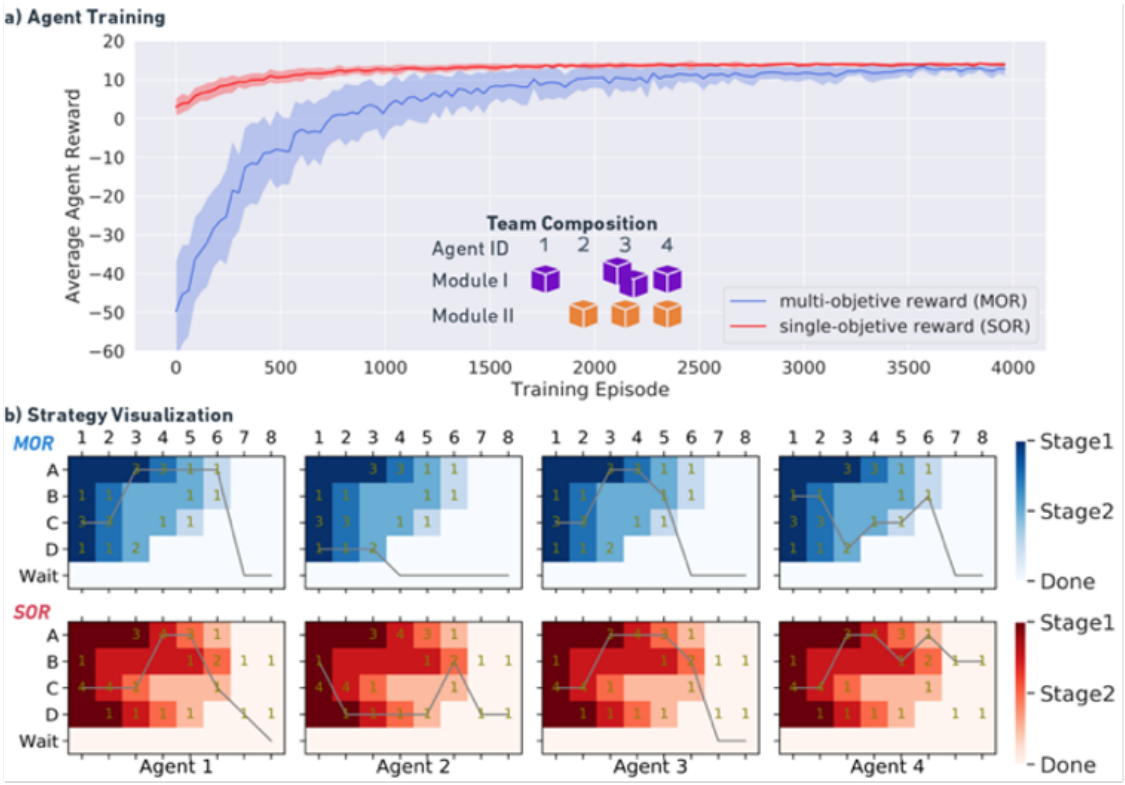


Figure 9.12: (a) Comparison of training curves and (b) strategy visualization of a team using MOR (blue) and SOR (red). Colors indicate task demands and stages, numbers indicate assigned modules, and lines represent decisions.

While team selection finds the best team composition out of the available resources (i.e., the available pool of agents and modules), next we explore an alternative strategy we refer to as team adjustment (with both removing and adding heuristics) that has the capability of refining the team to satisfy the desired task performance and to adapt the team to dynamically changing and unexpected situations. In Fig. 9.14, the team adjustment scenario is simulated to improve the team composition (initially the task is completed in 6 steps) to reach the maximum efficiency (that is completion in 4 steps). DwL identifies the module composition for agent 5 by analyzing the performance gap for each time step. The process iteratively advances until the newly added agent is found removable, i.e. the task cam be accomplished without that agent (who is always idling).

Comparing the two teams designed by selection and adjustment, the two processes end up with different team sizes and compositions. However, the two teams have the same number of modules of each type. This is because the initial team compositions are the same and the design heuristics

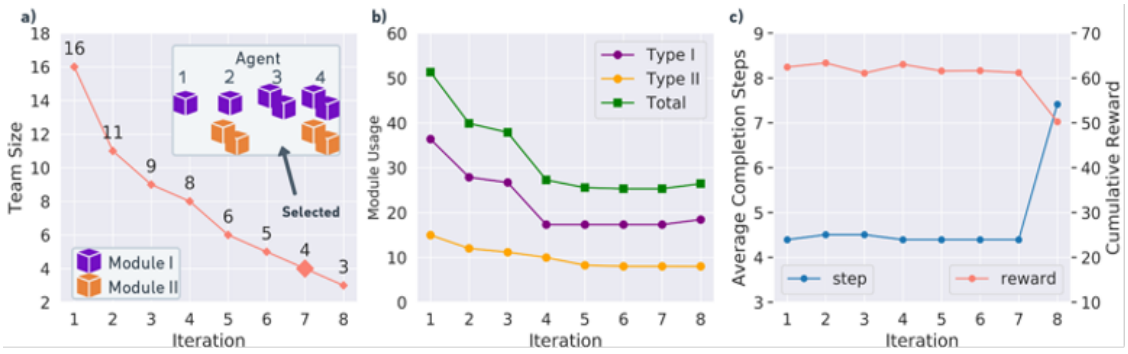


Figure 9.13: Team member selection process for each DwL iteration shown with (a) team size and the selected team, (b) module usage for both types, and (c) evaluated performance over 1000 realizations.

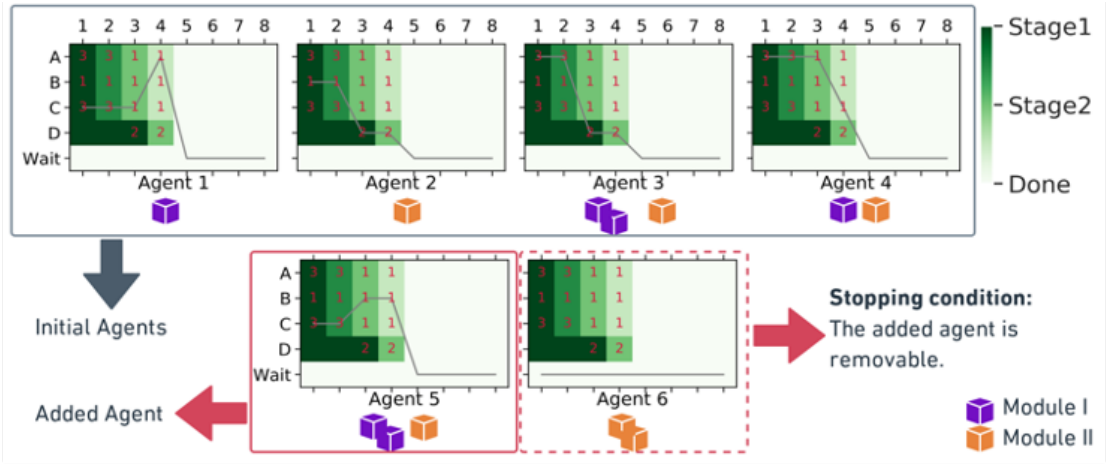


Figure 9.14: Team adjustment process using DwL, starting with the initial 4 agents (top) and visualizing the final team strategy.

favor usage of the same modules.

9.2.4 Conclusions

An intelligent anarchic learning paradigm was introduced to empower modular autonomous machines with enhanced flexibility and intelligent decision-making. The paradigm is designed for operations with multiple humans and multiple intelligent machines working as teammates. To combine strategic planning and adaptive team design, a design-while-learn framework using deep reinforcement learning and design heuristics was created and implemented to achieve both objectives simultaneously in an iterative process. The proposed approach and results contrast existing

methods that consider only fixed team compositions and are typically unresponsive to disruptions. Results from a simulated scenario suggest the effectiveness in training and design process and incentivize future research such as the development of explainable strategies and human-centric learning in smart manufacturing systems.

CHAPTER 10

Conclusions, Summary of Contributions, and Future Work

10.1 Conclusions

This dissertation contributes to the development of heterogeneous multi-agent teams (e.g., human-autonomy teams) for effective task allocation operations, particularly in generating decentralized coordination strategies for AI-powered autonomous teammates to collaborate with human decision-makers in various large-scale applications, ranging from disaster relief operations to manufacturing production floor planning, that involve complex task dynamics. This is challenging because these scenarios require multiple heterogeneous entities distributed dynamically, functionally, and spatially, while also considering the shared authority in strategic planning, the inclusion of human decision-makers, and unexpected events outside of the training envelope and learned expertise. Therefore, it is critical to comprehend such collaborative autonomous teammates before deployment by addressing three phases in the development pipeline of multi-agent teams, namely *learning*, *interpretation*, and *adaptation*.

Learning coordination strategies is the foundation to enable autonomous teammates to collaboratively achieve operational objectives. A decentralized artificial intelligence framework for heterogeneous teams is developed to learn dynamic task allocation decisions and maximize performance in complex operations through multi-agent reinforcement learning (RL). The proposed framework extends the state-of-the-art by explicitly modeling heterogeneity in task-handling capabilities and decision-making characteristics (including human factors) and by representing past experience in belief space for robust reasoning, capturing aspects of intelligence that produce effec-

tive collaborative teaming. By investigating the capability heterogeneity in the team composition and the risk tolerance in decision-making, results show that a well-designed heterogeneous team significantly improves operation efficiency, and including agents with higher risk tolerance in a team significantly improves the chance of success in sudden events. This decentralized decision-making framework is a foundational step that allows the modeling and training of heterogeneous agents with human characteristics, paving the way for teaming operations with actual human decision-makers. Another challenge that arises when involving humans is load management, as humans have limitations in handling heavy workloads in real practices (e.g., tele-operating 10 vehicles simultaneously or being actively engaged over hours). To further improve the decision-making framework, idling is modeled as an option for task allocation decisions, and reward functions are customized to encourage idling behavior and penalize unnecessary capability usage. This enhancement allows agents to reduce task load without compromising the completion efficiency of the operation and provides the opportunity to study precaution preparation.

Moreover, human-autonomy teaming for a disaster-relief operation is demonstrated in a game-based platform, where the collaborative autonomous teammates are trained by the proposed framework, and a human decision-maker is interfaced with a virtual adaptive user interface (AUI) summarizing available information and physiological sensors measuring workloads. The developed AUI helps mitigate cognitive overload by adaptively providing a suitable complexity level of information on other team members depending on operation stages and physiological measures. In this study, a dataset collected from 50 human participants is analyzed to better understand their decision-making behavior. An artificial human agent (i.e. an autonomous agent with human characteristics) is substituted with the real human decision-maker, allowing the team to successfully complete the mission, but human participants can never outperform the decisions recommended by the artificial human agent in trained scenarios. The advantages of having humans are their ability to react to sudden events and special knowledge in unforeseen situations. The results also show significant improvement in completion efficiency with the recommended decisions provided. This work allows human-autonomy interaction in reproducible scenarios and provides insight into

enabling autonomy to understand human strategies in teaming settings for achieving superior effectiveness of human-autonomy teaming in large-scale operations.

During the learning phase, the strategic behavior of the team emerges according to the common objectives used by multi-agent reinforcement learning. However, it remains unclear how strategies vary throughout different operation stages and how agents coordinate with each other to achieve the objectives. Performance metrics and behavior visualization approaches cannot sufficiently explain the underlying coordination dynamics. Therefore, the interpretation phase is necessary for autonomous teammates to facilitate the understanding of strategic behavior and humans with various preferences or roles. In this phase, A post-hoc interpretation metric is developed to quantify team coordination in different stress levels (defined by the remaining task demands of an operation). It first measures the pair-wise causality effect between agents using their decision time series data, then constructs a weighted network on decision causality for the team, and identifies highly coordinated sub-teams using a community finding technique. It is shown that RL-trained strategies can achieve a higher coordination level, compared to rule-based strategies, and applying a proper coordination pattern at the corresponding stages can avoid catastrophic failures.

Another situation where interpretation is critical is when team members first meet to collaborate or are unfamiliar with each other's preferences, especially when collaborating with human teammates or agents with changing preferences. Existing methods often assume the roles of teammates do not vary over time. To relax this assumption, interpreting the roles of teammates in collaborative tasks is essential. A Multi-agent Inverse Reinforcement Learning approach is proposed by incorporating the Theory of Mind to reason about how each individual might have modeled the intentions of others using the observations of their behaviors throughout the operation. The possibilities of the behavior intention are defined by a set of baseline profiles (reward functions). Experiments are conducted in a simulated search-and-rescue operation, and results show that ascribing the profiles of teammates not only accounts for imperfect knowledge about individual behavior and uncertain strategies of others but also enables a computationally efficient approach to compute the teaming strategy in a decentralized manner.

Real-world scenarios often differ from training scenarios. Unexpected or adversarial events that deviate from learned expertise are known as open-world novelty. Prompt adaptation to open-world novelty is an indispensable skill for autonomous teammates, especially in urgent situations where interventions (e.g., re-training) are not an option. It can only be achieved with the capability of understanding the interdependence between strategic behavior and task dynamics, potentially requiring the efforts discussed in the learning and interpretation phases. Additionally, multi-agent teams deserve extra attention in these challenges, as the breakdown of a single agent could lead to systemic failure. To this end, Bayesian Online Strategy Adaptation (BOSA) is proposed to enable multi-agent teams with the ability to address novelty by offline preparing a strategy pool, online inferring unusual events caused by novelty, and online adapting the most suitable strategy to deploy. The experiments show that BOSA can successfully recognize the correct strategy to deploy through the Bayesian reasoning capability and can instantly (with one-step observation) adapt to a sudden capability degradation novelty in the disaster relief operation with a strategy selection heuristic based on belief inclination. In addition, it has been validated that a single strategy can provide acceptable performance for multiple novel situations as operations scale, offering insights into more efficient offline preparation for strategy adaptation.

Besides the need for strategy adaptation, adapting the team composition altogether could help reach superior operational performance, within an allowable composition design space and given the resource constraints. This motivates better designing multi-agent teams for task allocation operations and fully automating the development cycle of teams composed of multi-attribute agents with strategic planning capabilities. Instead of finding the teaming strategy given a pre-defined composition, methods to simultaneously co-design team composition and team behavior are proposed. First, an iterative design and learning framework demonstrates the process of finding superior operations performance by selecting team members from an initial pool of agents or incrementally adjusting team compositions in disaster relief operations. Since this approach relies on heuristics to adjust the team composition, it provides clear explanations of why certain design changes are made. Furthermore, by drawing an analogy between chromosomes and multi-attribute

agents, a co-evolutionary approach is introduced to find multiple optimums with a voting mechanism indicating collective team fitness instead of evolving based on individual agent fitness. The co-evolutionary approach does not require additional knowledge or heuristics on composition modifications, but both team-level and agent-level performance metrics need to be carefully defined to guide the evolutionary process and eventually satisfy the objectives of the operation.

This dissertation not only demonstrates the potential for improved operational effectiveness through the development of coordination strategies for AI-powered autonomous agents with heterogeneous capabilities to collaborate with humans, understand operational dynamics, and adapt to unexpected scenarios, but also lays the groundwork for future research in intelligent multi-agent systems, further bridging the gap between theoretical models and practical, real-world applications.

10.2 Summary of Contributions

Following the conclusions, a summary of the main contributions to developing AI-powered autonomous teammates for task allocation operations is provided as follows.

The learning phase enables autonomous teammates to learn strategies for coordinating with human teammates and achieving operation objectives.

- A formulation with **explicit modeling of the dynamics between agent capabilities and task demands** into a decentralized decision-making framework to train teams with heterogeneous capabilities for complex multi-agent task allocation operations
- A methodology to quantitative evaluate **the combined effects of capability heterogeneity and risk aversion in agent decision models** on operational performance under unforeseen events
- An enhancement of the decision-making framework that achieves **multi-objective teaming behavior by efficient load management** without compromising completion efficiency.
- **A construction of a game platform** for human decision-makers interfaced with **a workload**

adaptive user interface to collaborate with autonomous agents in a disaster relief operation.

- **A dataset of physiological responses and decision behaviors** collected from a human-autonomy teaming subject study in the aforementioned platform.
- A methodology to investigate **the effect of adaptive information feeding and decision recommendations on the teaming performance with actual human decision makers** in immersive environments.

The interpretation phase empowers autonomous teammates to interpret teaming dynamics and explain the preferences of teammates.

- A generalized post-hoc interpretation approach to **measure causality among agents and identify coordinated sub-teams** using time series data of agents' decisions in decentralized multi-agent teams.
- A definition on **a quantitative metric for emergent team coordination** to interpret the patterns in different coordination strategies and team compositions across various operation stages.
- A development of **assistive AI that incorporates Theory of Mind reasoning** to explain the profiles and preferences of initially unknown human teammates and predict the behavior of others in collaborative tasks.
- **An application of reinforcement learning in algorithmic recourse** that provides explainable recommendations on how to change the input features of AI/ML models step by step to eventually reach a desired output with risk considerations.

The adaptation phase equips autonomous teammates with adaptability in strategically responding to evolving and novel situations. This phase also investigates the evolution of team composition along with behavior learning for designing MAS with superior effectiveness.

- An online strategy adaptation approach that enables multi-agent teams to **adapt to unexpected scenarios without re-training**, using offline preparation, online observations, and Bayesian reasoning.

- The proposal of strategy selection heuristics for **instant strategy adaptation under sudden capability degradation events** in multi-agent operations.

- A formulation of multi-agent task allocation operations, considering the **co-design of team composition and team behavior in strategic planning** for achieving superior performance.

- An iterative multi-agent team co-design framework that **alternates between continual behavior learning and heuristic-based composition design**.

- A co-evolutionary algorithm for **finding a diverse set of optimal team compositions** in multi-agent multi-attribute operations.

- **An application of multi-agent co-design in reconfiguration manufacturing systems** to achieve intelligent teaming among autonomous modular agents in task planning for production floor operations.

10.3 Future Work

Compared to previous coordination methods for multi-agent task allocation operations, the developed coordination strategies provide a wide range of techniques for collaborative strategic planning, load management, behavior interpretation, and strategy adaptation under novelty, offering the possibilities of developing future human-centered multi-agent systems across various domains. However, there are emerging challenges in these methods that require further investigation for future work.

The developed decentralized decision-making framework currently remains at the task planning level. It could be extended to incorporate agent-level motion planning and control actions at different planning hierarchies, providing a comprehensive testbed for teaming with any autonomy

stack. In addition, communication strategies play a significant role in human-autonomy teaming. Future work could consider strategies to transform human perception to machine languages, reconcile discrepancies in perceived information, and decide who to communicate with and when to communicate under the constraints of communication frequency and bandwidth. These efforts could further relax the burden of human agents to focus on other activities that require empathy, creativity, situation awareness, and critical thinking.

To achieve better human-autonomy collaboration and teaming performance, methodologies are needed to verify whether autonomy can raise situation awareness, understand human strategies, and realize evolving situations from human-demonstrated decisions. Additional experiments could be designed in the human-autonomy teaming game platform to study how autonomy can accommodate human teammates with different preferences and strategies. This becomes an essential step for developing human-centered autonomy that brings strategy learning, interpretation, and adaptation together.

To further leverage the developed coordination metric based on causality and sub-team identification to improve the creation of large-scale multi-agent systems, future work could include 1) analyzing the causality between agent behaviors and environmental dynamics, 2) monitoring and detecting unexpected coordination patterns when operational situations become unpredictable, 3) potentially simplifying the organization of team members with sub-teams, 4) improving the learning of coordination strategies based on causal effect among agents, and 5) investigating how communities (highly coordinated sub-teams) or leader-follower (one-directional causality) relations emerge in multi-agent operations. Moreover, besides explaining the profiles and intentions of teammates, the concept of the Theory of Mind reasoning could be extended to ascribe the states of the operational environment, allowing the prediction of the future dynamics for and the formulation of precaution strategies for out-of-distribution events to ensure adaptive, responsive, and resilient teaming performance.

In the initial investigation of multi-agent team co-design of composition and behavior, the iterative approach still requires domain knowledge to guide the co-design process. Although the

co-evolutionary approach does not rely on the designer's guidance, the evolution is still based on evaluated performance metrics. Given the computation power, the step-wise transition data when agents are interacting with the environment could be better utilized to infer the design changes along with the learning of coordination strategies. The potential approach could involve the application of transformer architectures that directly convert the state observations to co-design actions, with the "self-attention" mechanism that focuses the co-design decisions only on the important features in the representation space. The co-design study paves the way for future modular human-autonomy formations that can act optimally and re-configure adaptively in any complex task environment.

APPENDIX A

Safe Algorithmic Recourse by Risk-Aware Policies

A.1 Introduction

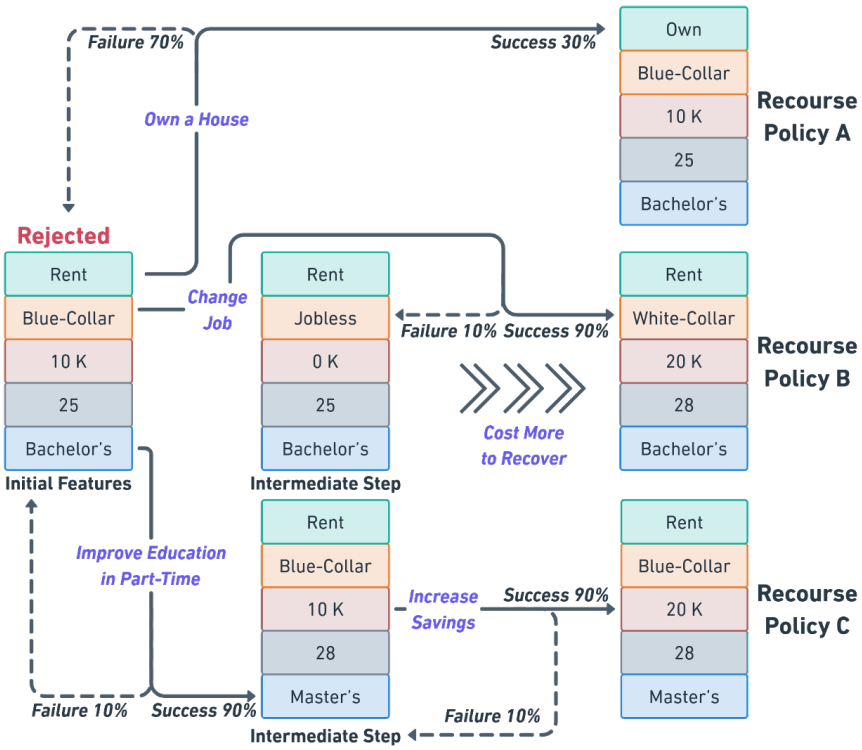


Figure A.1: Recourse policies for loan approvals. Policy A has only one feature change (low sparsity) but with a high failure rate; Policy B has the lowest expected cost but might result in a situation that costs more to recover; Policy C has a slightly higher expected cost than Policy B but lower variance in cost (risk), which can be considered as a safer Policy.

Machine learning (ML) models are increasingly being used to make decisions in a wide array of scenarios including healthcare [214], insurance premiums [215], and loan approvals [216].

Given the impact of ML models on society, the importance of algorithmic recourse has increased [217]. Algorithmic recourse refers to a computed recommendation provided to an end user that suggests specific changes they can make to convert an unfavorable outcome (e.g., loan rejection), into a favorable one. For a recourse to be helpful, the suggested change ought to be actionable; for example, one can change their savings balance but not their age. Existing recourse work has considered the cost of taking the recommended actions [145, 146, 218]. However, they do not consider the risk of higher costs. In this work, *risk* means the potential for higher costs during the recourse due to the possibility of reaching adverse states; this can happen due to uncertainties in action effects (not deterministic). The cost could be in terms of time required, effort, financial resources, etc. Without incorporating risks—which is ignoring uncertainties or only minimizing the expected costs—the recipient of an algorithmic recourse may be caught unaware and unprepared for situations with high costs. By offering recourse policies with risk measures, we can help people be aware of how much risk is involved in each policy and choose a safer one. The *recourse policy* here refers to the recommended actions for all possible states a person might encounter, as opposed to a single deterministic sequence of actions.

To further understand the need for risk considerations, let's look at the existing algorithmic recourse approaches that use *counterfactual explanation* (CE) methods to give recourse recommendations. CE methods find "the most similar instances to the feature vector describing the individual, that result in the desired prediction from the model" [144]. The assumption is that minimizing feature-space differences translates to a recourse that requires less cost to reach the desired outcome. Rather than providing a single vector of feature changes, recourse can also provide a series of CEs or a sequence of actions [218, 219] that incrementally change users' features to ultimately achieve the desired outcome. Some key desiderata to evaluate CEs are [143]: (1) *validity*: whether it gives the desired outcome, (2) *proximity*: how much the changes are measured by a distance function, (3) *sparsity*: how many features are changed, and (4) *realism*: how realistic recourse recommendations are for an individual, including the feasibility of actions. However, using CEs to find recourse policies does not necessarily result in a sufficiently ***safe*** recourse policy,

because they might ignore the risk of taking actions, which may (probabilistically) leave a person in a worse situation. Such a recourse policy may even be dangerous to suggest. For instance, asking a person to change jobs may result in them losing their current job and being jobless (as illustrated in Recourse Policy B in Figure A.1). Finding alternatives with lower risks but a slightly higher expected cost may be preferred by an individual. In the context of CE methods, this means that sometimes a more "distant" state (set of feature values) may be a better recourse target if the actions required to reach it carry less risk of higher costs.

To explicitly incorporate risk into algorithmic recourse, our work introduces the problem of computing *safe algorithmic recourse* (**SafeAR**). This has hitherto not been discussed in the literature on algorithmic recourse. The objectives of SafeAR are to suggest different recourse policies with different risk profiles and to empower the affected individual with risk-averse alternatives to decide for themselves.[1] Reinforcement learning (RL) methods can be used to compute such recourse policies. Typically, a policy in RL finds the best action given a (feature) state, which maximizes the expected reward (or minimizes the cost) and can incorporate uncertainty in cost and action effects. To account for risk, we incorporate the variance in costs during policy computation and connect risk-sensitive reinforcement learning ideas [147–149] with algorithmic recourse. Our contributions are:

- Develop the concept of SafeAR by highlighting the value of considering risks in algorithmic recourse, which existing recourse measures do not cover.
- Formulate algorithmic recourse problems as Finite Horizon Markov Decision Processes (MDPs) and propose a method (Greedy Risk-Sensitive Value Iteration, G-RSVI) to compute risk-aware policies for finite horizon MDPs
- Incorporate succinct measures of risk from financial literature to the assessment of algorithmic recourse; these measures are Value at Risk [150] and Conditional Value at Risk [151].

[1] SafeAR does not advocate for the policy with the lowest risk as it may have a higher average cost. The emphasis is to provide multiple recourse policies for individuals to choose from, some of which can be safer and more suitable based on their risk tolerance.

- Evaluate the policies with different risk profiles computed by G-RSVI on two real-world datasets (UCI Adult Income, German Credit), considering risk measures, sparsity, and proximity; demonstrate that the latter do not implicitly factor in risks.
- Initiate an investigation into gender disparities in terms of risk exposure in the aforementioned datasets.

A.2 Motivating Example

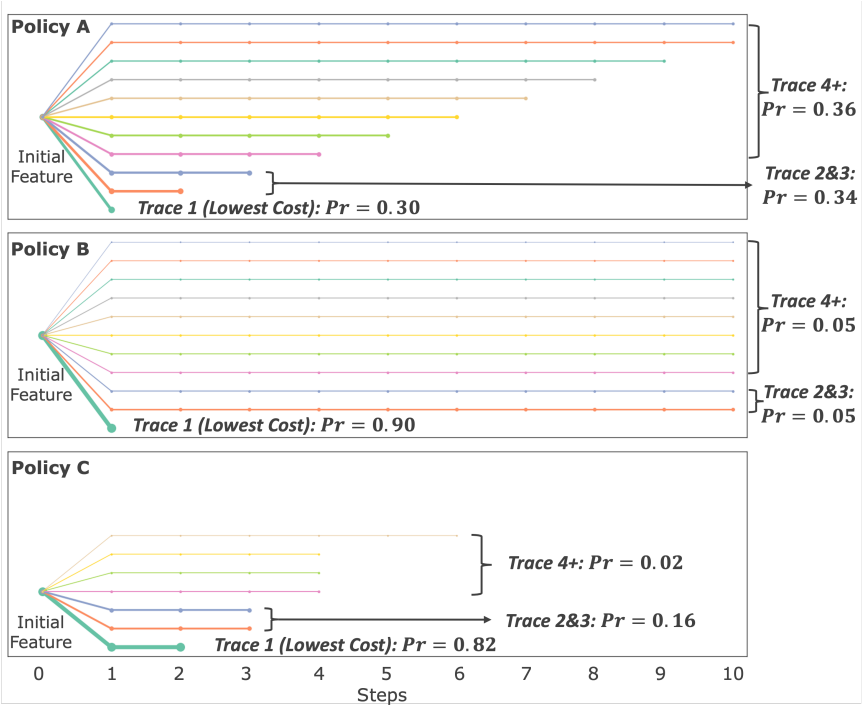


Figure A.2: Visualizing risks in three recourse policies from Figure A.1. The x-axis indicates the cost, and the line (outcome) thickness indicates the outcome probability.

To better illustrate the concept of SafeAR with risk-aware policies, consider the following motivating example. A company uses a trained black box ML model to determine loan approvals. The model uses a set of features of the loan applicant (housing, job, savings, age, and education) and initially rejects the applicant. In this recourse scenario, the action costs are in terms of time units (months), each action taken has a probability of success, and failure could transition into

a less favorable state. Let us look at three recourse policies that could be given to the applicant illustrated in Figure A.1:

- *Policy A: Nearest CE, Expected Cost 3.3*. This could be found by a recourse algorithm that optimizes for feature sparsity. It would require the applicant to *Own-a-House* (one feature change). This policy ignores the uncertainty in the applicant's ability to purchase a house within 1 month (time cost), and there is a 70% chance that the applicant would remain in the same state. Therefore, the expected time cost would be much more than 1 month.

- *Policy B: Risk-Neutral, Expected Cost 1.5*. This policy recommends the applicant to *Change-Job*, and doing so helps increase the savings and reach the desired outcome with 90% probability. However, there is a small chance (10%) that this action would result in losing their current job and becoming unemployed, from which the cost to recover would be much higher. The expected total cost when considering probabilities is the lowest among the three policies. If optimizing for expected cost alone, this policy would be found and might lead the applicant to a worse situation in which they require a high cost to recover from. This is the type of risk in a recourse policy that a user might want to know about and manage.

- *Policy C: Risk-Averse, Expected Cost 2.2*. This policy provides a safer policy to the applicant, where failures do not lead to a worse situation. The actions for this recourse are *Improve-Education-in-Part-Time* and then *Increase-Savings*. The risk associated with this policy is lower than Policy B, but it has a higher expected cost. Policy C might not be found by methods that minimize proximity, as improving educational background could be considered a larger change than getting a higher-paying job.

With the presence of uncertainties, different recourse trajectories (outcomes) might be encountered when following the recourse policy. Figure A.2 visualizes the probabilities of possible outcomes and their associated costs for this example. With the risk-averse Policy C, the applicant is able to receive the desired outcome in 3 time-steps (cost) with probability 98%, and the risk of it taking more than 3 is much less than Policy A or B, even if the expected cost is higher than Policy

B. Computing such diverse policies in terms of risk and surfacing the risk information to empower the affected individual is the motivation behind SafeAR.

A.3 Related Work

Existing algorithmic recourse methods [144] can be grouped into three categories: one involves finding the nearest CEs as the smallest changes to the individual's feature vector. Solutions focus on *proximity* [220], *sparsity*, and *diversity* [221–224] using multi-objective optimization [225] and decision trees [226]. Also, generative algorithms [227, 228] are used to ensure *plausibility*, by generating CEs within data distributions. These methods do not give a sequence of actions or policy to follow and have no mention of risk.

In the second category, recourse is achieved by recommending a sequence of actions [229] or by providing a path over the feature-space along dense regions of the data manifold [218] considering *feasibility* and *actionability*. Methods also incorporate *causality* through structural causal models (SCMs) [230] to explicitly model inter-variable causal relationships [231] and provide an ordered sequence of CEs [219, 232]. Lastly, robust recourse methods [233, 234] address the issues for data changes and model parameter shifts. None of these methods consider the risk of higher costs due to the probability of adverse outcomes.

For computing risk-aware recourse policies, we turn to the reinforcement learning (RL) literature. RL methods can provide recourse policies that consider uncertainties in transitions when taking recourse actions. There is existing work that models the recourse problem as MDPs [235]. "ReLAX" [236] generates recourse plans by deep reinforcement learning but under deterministic feature transitions, ignoring uncertainties and thus risk. FASTAR [146] presents a framework that translates an algorithmic recourse problem into a discounted MDP and demonstrates comparable recourse performance as CE methods. Although FASTAR models uncertainties, it only optimizes for the expected cost and does not incorporate any risk measures. Our first method for SafeAR (G-RSVI) computes recourse policies by considering both the expected cost and the risk of higher costs.

One way of measuring risks in cost in RL is through the variance in the total cost (over all steps) [237, 238]. There are also other RL methods that factor in risks in policies [147, 239–241]. To communicate the idea of SafeAR in this work, we use a modification of value iteration (G-RSVI) to incorporate the cost-variance trade-off into the computation to get risk-averse policies. MDPs can naturally incorporate action costs, probabilistic action dynamics, action feasibility, and causal constraints. These can all be personalized to the recipient, as properties like action costs can be unique to each person. To the best of our knowledge, SafeAR is the first attempt to connect the literature on *risk-sensitive* RL to algorithmic recourse.

A.4 Safe Algorithmic Recourse

A.4.1 Algorithmic Recourse Problem Statement

Let $f : \mathcal{X} \to \mathcal{Y}$ be a decision function operationalized by a ML algorithm or model, where $x \in \mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_D$ is the set of instances described by D features of an individual, and $\mathcal{Y} = \{y^-, y^+\}$ are the unfavorable and favorable decision outcomes, respectively. An individual with features x_o initially gets an unfavorable outcome $f(x_o) = y^-$, and the general objective of algorithmic recourse is to find actions resulting in a path $x_o, \ldots, x^*$ that leads to final feature instance x^* so that $f(x^*) = y^+$. Our work is agnostic to the type of ML model f and only requires the model outputs to be categorized into unfavorable outcomes y^- and favorable outcomes y^+. For simplicity of discourse, we use a binary classifier for f.

A.4.2 Risk-Aware Recourse Policies Using Finite Horizon Markov Decision Processes

To compute SafeAR recommendations, we frame the problem as solving a finite horizon Markov Decision Process (MDP), defined as a tuple of $\langle S, A, T, R, H \rangle$.

A.4.2.1 States (S)

S is a set of all possible states for individuals in the recourse. Each of the states maps to one instance ($x \in \mathcal{X}$) in the combined feature space (input space) of the decision model f. For a valid state space, there must exist a mapping $g := S \to X$, where $\forall s \in S, \exists x \in X$ such that $g(s) = x$. In this work, we keep the mapping g as one-to-one, meaning the state space is equivalent to the feature space. However, the state space S can be *richer* than the feature space $\mathcal{X}$ because the states and actions for recourse can involve more or different features than the ones used in the decision model f. For example, "resting heart rate" can be a feature in a health insurance premiums calculator f, but "average calories burned" is not. However, the latter may be in the recourse state, as it is directly affected by actions (e.g., *Exercise*), and in turn can have a causal effect on "resting heart rate". Using only the same features as in f may not be adequate for computing recourse policies, as they may not cover the states and actions that a person actually has to change during the recourse. This gives us a reason to expect separate action, transition, and cost models for recourse, rather than assuming they can be extracted from the data used in f. Such formulation also enables the recourse to provide personalized recommendations to individuals, as advocated for in [217].

A.4.2.2 Actions (A) and Transitions (T)

For the state space S, we have a set of feasible actions $a \in A$. The effect of an action can change multiple features. The features can be categorized into three types [242]: 1) immutable features (e.g., birthplace), 2) mutable and actionable features (e.g., occupation, bank balance) that define the action space of the recourse, 3) mutable but non-actionable features (e.g., credit score) that cannot be directly modified by an individual. Mutable features can be modified as a consequence of changing other features. The state transition model would need to capture causal relationships between features and ensure the realism of recourse. The state transition model $T := p(s'|s, a)$ is defined as the transition probability between two states ($\{s, s'\} \in S \times S$) given the action a.

A.4.2.3 Rewards (R)

$R := r(s, a, s'; f)$ is the reward or cost incurred by reaching state s' by performing an action a at a state s. "Reward" and $r(.)$ are the typical terms and notations used in RL literature, but rewards can be positive or negative (cost). We will henceforth use "cost" in this work since we are focusing on the recourse cost to the recipient, i.e. $r(s, a, s'; f)$ tells us the cost incurred to the recipient when the transition (s, a, s') occurs during the recourse. Additionally, when the ML model f gives the favorable outcome in a state ($f(s) = y^+$), then no more actions are needed in the recourse. To capture this, we add a zero-cost action in all favorable (goal) states, transitioning to the same state.

As for the real-world semantics of the cost, it can be a combined measure of multiple factors such as elapsed time, material expenses, opportunity cost, etc. The cost may be averaged across a group or tailored for each person, which requires domain knowledge—so do the feasible actions and the transitions. CE methods such as DiCE [223] and FACE [218] also require domain knowledge to design distance (cost) functions, where the function $r(.)$ can be defined in terms of how much the state changes by an action using *sparsity* of feature changes and *proximity* of the recipient's state changes over pre-defined distance functions.

A.4.2.4 Horizon (H) and Recourse Policies

Horizon H is the maximum number of steps in the finite horizon MDP, and $h := [1 : H]$ is the step number over the horizon. A recourse policy is the same as an MDP policy, expressing how to act in each state of each step in the horizon to get to a favorable state. This is formalized as $\pi = (\pi_1 \dots \pi_H)$, where $\pi_i := S \to A$ maps each state to an action for each step i.

SafeAR Methodology

We present a method to compute risk-averse recourse policies and measures to evaluate the risk for SafeAR.

A.4.3 Greedy Risk-Sensitive Value Iteration

We present a greedy algorithm to compute risk-averse policies by incorporating cost variance. We first define $\hat{R}_h^\pi(s) = \sum_{i=h}^{H} r(s_i, a_i, s_{i+1}) | (s_h = s), \pi$ as the total recourse cost accrued over the horizon H from a rollout obtained by following a policy π starting at state s and step h. Since outcomes of actions are probabilistic, the total recourse cost is a distribution. In risk-neural settings, the recourse policy π maximizes the expected total cost $\mathbb{E}[\hat{R}_h^\pi(s)]$ or the mean value $\mu[\hat{R}_h^\pi(s)]$. G-RSVI also considers the variance in the total recourse cost to manage risk and seeks to find a policy π to maximize the following value function for each state s starting at the first step $h = 1$:

$$V_1^\pi(s) = \mu(\hat{R}_1^\pi(s)) - \beta \cdot \sigma(\hat{R}_1^\pi(s)). \tag{A.1}$$

We denote $V_h^\pi(s)$ as the risk-sensitive value of state s in step h by following policy π. $\beta \geq 0$ is the tuning parameter that represents the risk profile, and a higher value means more risk averse. When $\beta = 0$, the problem reduces to finding the policy with the least expected cost only, which is the standard optimization objective in MDPs. Here, σ returns the standard deviation of the total cost, and σ^2 returns the variance. In G-RSVI, we optimize Equation A.1 by greedily maximizing $V_h^\pi(s)$ at each step starting from the end step H and moving backward to the first step. At each step h, the action is selected to maximize the risk-sensitive value using:

$$V_h = \max_a \mu[r(\cdot) + V_{h+1}] - \beta\sigma[r(\cdot) + V_{h+1}], \tag{A.2}$$

where V_{h+1} is the value computed in the previous step. The risk-sensitive action value or Q-value $Q_h(s, a)$ at step h is defined as:

$$\begin{aligned} Q_h(s, a) = \mathbb{E}_{s'}[r(s, a, s') + V_{h+1}(s')] \\ - \beta\sigma[r(s, a, s') + V_{h+1}(s')]. \end{aligned} \tag{A.3}$$

Algorithm 6 G-RSVI

Input: recourse MDP$\langle S, A, T, R, H\rangle$, ML model f
Parameters: risk aversion level $\beta \in [0, \infty]$

1: $V_{H+1}(s) \leftarrow 0, \forall s \in S$
2: **for** step $h = H, H-1, \ldots, 1$ **do**
3: **for** each state $s \in S$ **do**
4: **for** each action $a \in A$ **do**
5: $r(s') \leftarrow$ get reward $R(s, a, s'; f)$
6: $p(s') \leftarrow$ get transition probability $T(s'|s, a)$
7: $\mu \leftarrow \sum_{s'} p(s')[r(s') + V_{h+1}(s')]$
8: $\sigma^2 \leftarrow \sum_{s'} p(s')[r(s') + V_{h+1}(s') - \mu]^2$
9: $Q_h(s, a) \leftarrow \mu - \beta\sigma$ (Equation A.3)
10: **end for**
11: $V_h(s) \leftarrow \max Q_h(s, \cdot)$
12: $\pi_h(s) \leftarrow \text{argmax}\, Q_h(s, \cdot)$
13: **end for**
14: **end for**
15: **return** recourse policy $\pi_h(s)$

If only optimizing the expected reward, this procedure would find the optimal policy because the optimal sub-structure assumption for dynamic programming holds. However, G-RSVI does not guarantee to find the optimal policy for Equation A.1. It does, however, provide one straightforward way to incorporate risks into recourse policy computation, and it completes computation with a single sweep over the state and horizon space. There are a variety of heuristic methods one can use with different trade-offs to compute risk-aware policies. In this work, to focus on the exposition of the concept of SafeAR, we limit our approach to discrete states, discrete actions, and finite horizon MDPs.

Our G-RSVI algorithm is shown in Algorithm 6. We compute the policy by sweeping backward from the last horizon step (Line 2). For all state-action pairs at each step, the action values $Q_h(s, a)$ are computed by Equation A.3 (Lines 5-9). The best action for each state in each step is then chosen by the one with maximal $Q_h(s, a)$. It also gives us the state value $V_h(s)$ and the policy for each step (lines 11, 12). Other ways of scoring values and selecting actions can be used in our algorithm. For example, one can also penalize Lower Partial Standard Deviation (LPSD) [243] (Appendix A.8.8) of lower-than-average values (negative) to avoid higher-than-average costs, as

greater negative values indicate higher costs. Equation A.3 helps us compare against risk-neutral policies that optimize for expected value only, by setting $\beta = 0$. We leave the analysis of different risk-sensitive algorithms for recourse to future work.

A.4.4 Risk Measures for Recourse Policies

To evaluate the risk associated with a recourse policy, we propose the following measures.

A.4.4.1 Success Rate (ρ_H):

It estimates the probability of success within the horizon H by following the recourse policy. For example, $\rho_5 = 0.9$ means a favorable outcome state will be reached within 5 steps 90% of the time. ρ_H is not equivalent to *validity*, which only determines whether a feature instance with a favorable decision exists. ρ_H is affected by the uncertainty of action outcomes in the recourse policy.

A.4.4.2 Mean-Variance Cost ($\mu_{cost}, \sigma^2_{cost}$):

It computes the expected value and variance in the total cost of following recourse policies. Since the distribution of costs is not necessarily Gaussian, these statistics can be misleading or hard to interpret. Hence, we propose additional measures.

A.4.4.3 Value at Risk (VaR$_\alpha$):

VaR [150] is to provide a succinct probabilistic guarantee on the recourse policy cost. We evaluate VaR [150] of the recourse cost to answer the question "What is the highest cost at a given level of cumulative probability (confidence level)". For example, VaR$_{95} = 5.6$ means that with 95% probability, the recourse cost is at most 5.6. Formally, assuming the total cost of recourse x_c is the value of a random variable X_c with a cumulative probability distribution $F_X(x_c)$, under

confidence level $\alpha \in [0, 1]$ VaR_α is computed as:

$$\text{VaR}_\alpha(X_c) = \min\{x_c | F_X(x_c) \geq \alpha\}. \tag{A.4}$$

A.4.4.4 Conditional Value at Risk (CVaR_α):

CVar [151] is a complementary measure to VaR and tells us the expected worst-case cost when the cost exceeds the threshold given by VaR_α value. For example, $\text{CVaR}_{95} = 8.4$ means that when the cost exceeds the 95-percentile cost, the average cost for those cases is 8.4. It is computed as:

$$\text{CVaR}_\alpha = \mathbb{E}[x_c | x_c > \text{VaR}_\alpha]. \tag{A.5}$$

A.5 Experimental Results

Dataset	Policy	$\rho_{H=12}$	$(\mu_{cost}, \sigma^2_{cost})$	VaR_{80}	CVaR_{80}	VaR_{95}	CVaR_{95}	Spars.	Proxi.
Adult Income ($n = 25923$)	$\beta = 0$	0.994	(**3.49**, 1.23)	3.81	6.31	4.76	7.53	**2.09**	**2.87**
	$\beta = 0.25$	0.994	(3.51, 0.89)	**3.64**	**6.10**	4.46	7.43	2.16	3.06
	$\beta = 0.5$	0.993	(3.59, **0.77**)	3.66	6.11	**4.44**	**7.40**	2.21	3.18
(Example Instance)	$\beta = 0$	1.000	(**4.63**, 1.86)	5.80	8.54	6.80	9.80	3.92	**3.92**
	$\beta = 0.5$	1.000	(4.79, **0.13**)	**4.80**	**6.80**	**4.80**	**6.80**	**3.00**	4.86
	$\beta = 0.75$	1.000	(4.79, **0.13**)	**4.80**	**6.80**	**4.80**	**6.80**	**3.00**	4.86
German Credit ($n = 281$)	$\beta = 0$	1.000	(**1.65**, 0.48)	1.96	3.66	2.63	4.56	**1.26**	**1.33**
	$\beta = 0.25$	1.000	(1.67, 0.35)	**1.87**	**3.51**	2.51	4.50	1.34	1.43
	$\beta = 0.5$	1.000	(1.70, **0.30**)	1.90	3.56	**2.48**	**4.40**	1.40	1.50
(Example Instance)	$\beta = 0$	1.000	(**2.48**, 3.49)	**4.00**	6.13	7.00	8.33	**1.00**	**1.00**
	$\beta = 0.5$	1.000	(2.81, 1.19)	**4.00**	**5.44**	**5.00**	**6.00**	**1.00**	2.00
	$\beta = 0.75$	1.000	(3.87, **0.65**)	4.40	5.50	5.40	6.67	2.00	3.00

Table A.1: Evaluating recourse policies with different risk-aversion levels β and horizon $H = 12$ for AID, GCD, and two example instances sampled from the datasets. n denotes the number of instances used for evaluation. For each risk-aversion level, we compute a recourse policy and apply it to each instance (initially with the undesired outcome) in the dataset. Due to the uncertainties in action outcomes, we estimate the average of each measure using simulated 100 trials. Then, the average across all instances for each metric is computed. The best metric values among the policies are highlighted in bold.

Motivated by the datasets used in the algorithmic recourse literature, we evaluate our method on the following two datasets: Adult Income Dataset (AID) (32561 data points) [244] and German Credit Dataset (GCD) (1000 data points) [245] and show how risk measures vary with different

recourse policies. In AID, the recourse is to help individuals earn an income greater than 50,000. In GCD, the recourse is to help get a loan approval by reaching a good credit standing. Here we consider the version of GCD [246] with 9 features. To process the datasets for G-RSVI, we convert continuous feature values into discrete values (details included in Appendix A.8.1). We then train random forest classifiers for both datasets. Dataset features, feature state dimensions, and classifier accuracies are reported in Table A.2. G-RSVI does not access the dataset when computing the policy and only uses the trained ML model to indicate whether the desired outcome is received.[2]

Dataset (#Feat.)	**Immutable Features**	**#States**	**ML Model (Accuracy)**
AID (8)	Gender, Race, Marital Status	57600	Rand.Forest (0.81)
GCD (9)	Gender, Purpose, Credit Amount	147456	Rand.Forest (0.76)

Table A.2: Dataset Overview

Dataset	**Policy**	$\rho_{H=12}$	$(\mu_{cost}, \sigma^2_{cost})$	**VaR$_{80}$**	**CVaR$_{80}$**	**VaR$_{95}$**	**CVaR$_{95}$**	**Spars.**	**Proxi.**
Adult Income (Female, $n = 9824$)	$\beta = 0$	0.988	(**4.56**, 1.41)	4.76	7.07	5.70	8.00	2.58	**3.77**
	$\beta = 0.25$	0.987	(4.57, 1.18)	**4.64**	**6.95**	**5.48**	**7.84**	**2.55**	3.87
	$\beta = 0.5$	0.985	(4.61, **1.11**)	4.66	6.99	5.51	7.89	**2.55**	3.93
(Male, $n = 16099$)	$\beta = 0$	0.997	(**2.84**, 1.12)	3.27	5.79	4.30	7.26	**1.79**	**2.32**
	$\beta = 0.25$	0.997	(2.87, 0.72)	**3.08**	5.51	3.95	7.15	1.93	2.56
	$\beta = 0.5$	0.998	(2.98, **0.57**)	3.10	**5.48**	**3.92**	**7.08**	2.01	2.73
German Credit (Female, $n = 103$)	$\beta = 0$	1.000	(**1.72**, 0.56)	2.08	3.70	2.84	4.66	**1.25**	**1.36**
	$\beta = 0.25$	1.000	(1.75, 0.38)	**2.00**	**3.61**	2.67	4.63	1.33	1.47
	$\beta = 0.5$	1.000	(1.79, **0.33**)	2.02	3.64	**2.59**	**4.51**	1.41	1.57
(Male, $n = 178$)	$\beta = 0$	1.000	(**1.61**, 0.43)	1.89	3.64	2.61	4.62	**1.27**	**1.32**
	$\beta = 0.25$	1.000	(1.62, 0.37)	**1.81**	**3.50**	2.41	4.42	1.35	1.41
	$\beta = 0.5$	1.000	(1.65, **0.29**)	1.84	**3.50**	**2.33**	**4.29**	1.40	1.46

Table A.3: Evaluating recourse policies for gender for Adult Income and German Credit datasets; the same evaluation procedures are followed as Table A.1. The male group is exposed to less risk under the policy with the same risk-aversion level.

A.5.1 Transitions and Rewards

We use qualitative assumptions (domain knowledge) on relative differences in action costs and success likelihood to define the action costs $r(\cdot)$ and transition model $p(\cdot)$. Similar to FASTAR

[2]All supplemental materials (appendices and code implementations) are available through arxiv.org/abs/2308.12367.

[146], we assume *improve-education* or *improve-skill* actions would lead to an age increase as causal constraints, and we treat "Age" as a mutable but non-actionable feature. These two actions require more time and effort, and therefore the action cost would be larger than other actions such as to *increase-work-hours*. The transition probabilities are heuristically set by domain knowledge. For example, the probability of earning a Ph.D. degree is lower than earning a Bachelor's. For more details on the model values, we refer the reader to Appendix A.8.2 for an exhaustive list of model transition probabilities and costs. Results from a different model using the same qualitative assumptions are also provided in Appendix A.8.5 to show the G-RSVI method and results are not specific to a single model.

A.5.2 Baselines

To our knowledge, our work is the first to address risks in algorithmic recourse. Among the existing recourse approaches, CE methods do not naturally allow probabilities into the formulation, and FASTAR [146] formulates recourse problems as MDPs *and* allows for stochastic transitions. FASTAR sets rewards in terms of distance measures between states. No matter what reward function is used—either distance-based or user-defined cost—and how transition probabilities are defined—either extracted from a dataset or tuned domain knowledge—FASTAR only seeks to find the recourse policy that maximizes the expected total rewards (risk-neural). This is what a standard algorithm for MDP (value or policy iteration) would find. In our experiments, the policy that maximizes expected total reward corresponds to the risk-neutral policy ($\beta = 0$), and this is the baseline which risk-averse policies compare against. We select $\beta = 0.25, 0.50, 0.75$ for generating risk-averse recourse policies, and higher β indicates higher risk-aversion.

A.5.3 Performance Evaluation

Table A.1 reports the risk measures for each experimental setting. The horizon is set to 12. We also present sparsity (L_0 distance) and proximity (L_0 distance for nominal features + L_1 distance for ordinal and numerical features) between initial and final states. All measures are averaged over

the entire dataset, as well as the measures for two example instances (sampled from $\mathcal{X}$). Recourse policies are computed using the same cost and transition functions for all instances in the dataset. In the results, we see that for both datasets, more risk-averse policies (higher β values) can provide recourse with less variance in cost σ^2_{cost} but often require a higher mean cost μ_{cost}. For the same α confidence level, risk-averse policies give lower costs in VaR and CVaR than risk-neutral policies. Also, for the example instance in GCD, variance in cost $\sigma^2 = 0.65$ at risk-aversion level $\beta = 0.75$ is significantly lower than the $\sigma^2 = 3.49$ at $\beta = 0$. For the example instance in AID, increasing risk-aversion to $\beta = 0.75$ would not find a different policy than $\beta = 0.5$, which can happen if the same relative ordering of state values $V_h(s)$ is found at each step. In the results, we observe that low sparsity and proximity do not correspond to risk-averse policies, meaning optimizing for them would not necessarily factor in risks.

A.5.4 Visualizing Risks in Recourse Policies

In Figure A.3, we use our policy-risk visualization for a set of policies from the GCD dataset. For each policy, we show the most probable outcomes (rollouts), the length of each trace corresponds to the total cost, and the thickness of each trace corresponds to the probability of the outcome. This approach visualizes the variability in cost, which can help a person get an intuition of their risk in addition to the recommended actions.

A.5.5 Exploring Risk Disparity

Inspired by prior work that investigated recourse fairness [247–250], we now look at the disparity that may exist in risk measures across two gender groups (male and female) provided in AID and GCD. Table A.3 reports the risk measures (averaged) for female and male groups in both datasets. The p-values for statistical significance of the difference between the groups are provided in Appendix A.8.4. We define disparity in VaR between the two gender groups by following the same recourse policy computed with a risk-aversion level β as: $\Delta\text{VaR}^{\beta}_{95} = |\text{VaR}^{\text{Female}}_{95} - \text{VaR}^{\text{Male}}_{95}|$. The disparity for other measures is similarly computed.

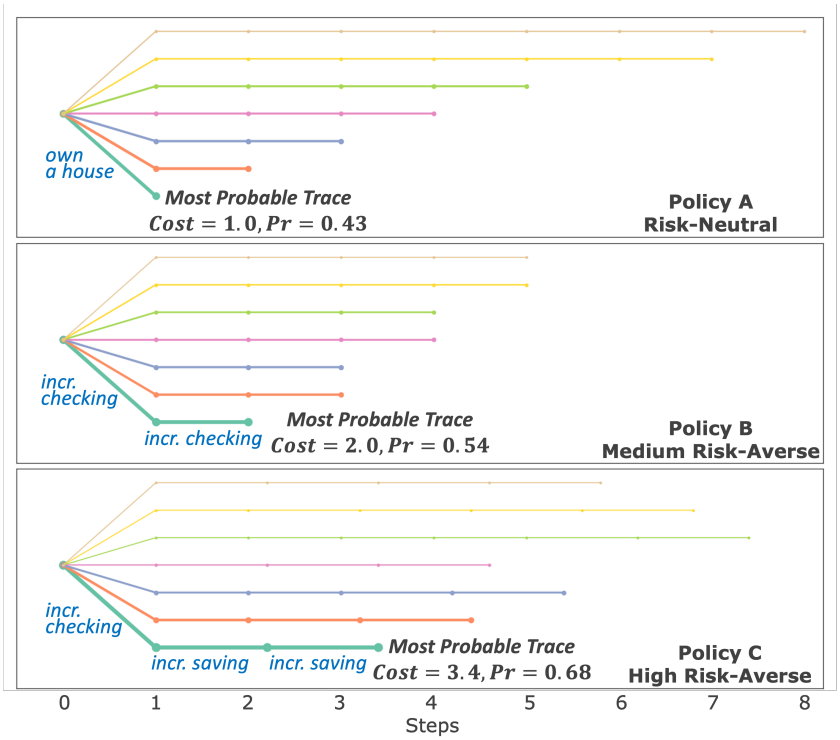


Figure A.3: Policy visualization Example in German Credit.

All measures are in favor of the male group for both datasets, meaning that for the policy with the same risk-aversion level given to females and males, we expect females would get higher variance in cost (σ^2_{cost}), higher cost at the VaR confidence level of $\alpha = 80, 95$, and higher costs in the expected worst case scenarios (CVaR) for those confidence levels. In AID, when increasing the risk-aversion, the disparity of risk measures between two groups becomes *larger*. We observe $\Delta\text{VaR}^{\beta}_{95}$ increases from 1.4 to 1.59 as β increases, and similar trends are observed for the difference in σ^2_{cost} and CVaR in AID. This trend indicates that the more we want to achieve risk-aversion, the greater the disparity in risk exposure between the two gender groups in AID. However, in GCD, the difference in σ^2_{cost} and $\Delta\text{VaR}^{\beta}_{80}$ do not consistently increase with increased risk-aversion. The disparity between males and females across all measures of risk still exists in GCD. We recall that the same action costs and transitions are used for both males and females. The only difference is the decisions made by the model f for different groups, which affects the number of steps to reach the favorable state ($f(x) = y^+$). This is something that recourse providers may want to keep in

mind, and it motivates further discussion on risk disparity in algorithmic recourse.

A.6 Discussion and Conclusions

The motivation behind our Safe Algorithmic Recourse (SafeAR) is to offer recourse policies with different risk profiles. This enables affected individuals to be aware of the risks and helps them make an informed decision based on their risk tolerance. We connect ideas from risk-sensitive reinforcement learning with the algorithmic recourse literature and propose an algorithm G-RSVI that can provide risk-averse recourse policies for individuals with different risk profiles. In our experiments with the AID and GCD datasets, we showed that the recourse policies generated by G-RSVI were better in terms of the risk measures as compared to the existing risk-neutral approaches. The policy risk was evaluated through cost-variance, VaR, CVaR, and success rate. In addition, we observed that policies with better sparsity and proximity scores need not correspond to risk-averse policies. Lastly, in our experiments, we observed discrepancies between gender groups in risk measures for the same risk-aversion setting, which motivates further studies on recourse fairness in terms of risk exposure.

A.7 Ethical Statement

SafeAR would work best with personalized action costs and action success likelihoods for an individual (no personal data was collected or used in this work). If done that way, it would effectively mean asking for personal information. However, the personal data can be deleted after computing the recourse policy, as SafeAR only requires individual action-model data for computation. Also, the black-box ML model does not need information on personal models and preferences on actions and transitions.

A.8 Supplemental Materials

A.8.1 Pre-processing Datasets

To prepare the Adult Income Dataset (AID) [244], we use the same features as DiCE [223] and follow the same process as [251] used by DiCE. Income <50K is considered as unfavorable, and >50K is favorable, indicated as $\{0, 1\}$ respectively. In addition, there are two features with integer values: Age and Hours-per-Week (Hrs/Week), and we categorize them into ordinal values as the requirement for our approach, G-RSVI. Similarly, we categorize Credit-Amount (Credit#), and Duration in the German Credit Dataset (GCD) [246]. The resultant feature lists and feature categories/levels are reported in Table A.4.

Adult Income Feature Values	**Target: Income={0,1}**
Age	<20, <25, <30, <40, <50, older
Education	School, HS, Bachelors, Masters, Doc.
Hrs/Week	Less, Part-Time, Full-Time, Over
Workclass	Gov., Self, Private, Other
Occupa.	Prof., WhiteCol, Service, BlueCol, Sales, Other
Marital	Single, Married, Divorced, Separated, Widowed
Race	White, Other
Gender	Female, Male
German Credit Feature Values	**Target: Risk={0,1}**
Age	Student, Young, Adult, Senior
Skill	Little(Non-Res), Little(Res), Skilled, Manage.
Savings	None, Little, Moderate, Rich
Checkings	None, Little, Moderate, Rich
Duration	<1yr, <2yr, <3yr, More
Housing	Free, Rent, Own
Purpose	TV, Education, Equip., Car, Business, Other
Credit#	Low, Med, High, High Plus
Gender	Female, Male

Table A.4: The feature list and discrete values for both datasets. Bolded features are immutable. The other features are mutable, and Age is non-actionable.

A.8.2 Transition and Cost Functions

We direct the reader to Table A.5 in which we report the action space, action costs (rewards), and the corresponding transitions in terms of action success likelihood used for AID and GCD

evaluation in Section A.5. When an action fails, we just assume the user stays in the same state. We assume each action can only change one feature, but Age will increase when taking Impr-Edu (improve education) or Impr-Skill (improve skill). For ordinal features, we only allow the state transition to increase by one level; for example, we do not allow Hrs/Week to jump from "Part-Time" to "Over" (overtime). The costs and probabilities are set based on qualitative assumptions and basic domain knowledge; this is used to determine which actions have higher costs than others. Ideally, these recourse models (action cost and transitions) should be designed for each individual based on their situation. For the purpose of evaluating and showing different recourse policies with different risk-aversion levels, we use the same cost and transition models for the entire dataset. We show how our method produces risk-aware policies for specific instances in Section A.8.10.2, and another set of models preserving similar qualitative assumptions is evaluated in Section A.8.5.

Adult Income		
Action Space	Action Costs $r(a)$	Transitions (Prob. of Success) $p(s')$
Impr-Edu	2.0	HS:1.0, Bachelors:0.9, Masters:1.0, Doc. 0.8
Incr-Hrs	0.8	1.0
Work-Gov	1.1	0.7
Work-Self	1.0	0.5
Work-Private	1.2	0.9
Job-Prof	1.0	0.5
Job-WhiteCol	1.0	0.5
German Credit		
Impr-Skill	1.5	Skilled:0.8, Manage:0.8
Incr-Savings	1.2	0.95
Incr-Checkings	1.0	0.7
Incr-Duration	1.0	1.0
House-Free	1.5	1.0
House-Rent	1.2	0.7
House-Own	1.0	0.4

Table A.5: Action Cost and Transition Models; If the action fails, the state stays the same.

A.8.3 Incorporating Counterfactual Explanations

Recall that CE methods find the final feature instances or a path of feature instances to achieve recourse. CE methods do not consider feature change uncertainties and transition probabilities, so

our method (G-RSVI) cannot be directly compared with CE methods. However, as an approach to incorporate CE methods, we can use the final feature instances provided by CE methods as goal or objective states in our MDP; these are treated as the only favorable outcome states and then we compute the policies by G-RSVI. Here, we first generate 10 CEs using DiCE [223] (denoted as DiCE10) and apply G-RSVI to compute the risk-neutral policy that leads to DiCE10 goal states with favorable outcomes (DICE10$_{\beta=0}$). The cost and transition functions are still required as they model the costs incurred by taking actions and uncertainties of feature changes for users. The performance evaluation is reported in Table A.9. We notice that using DICE10$_{\beta=0}$ has the lowest sparsity, but other measures are outperformed by policies that explore the full state space, especially the recourse success rate. These results imply that CE methods could potentially prevent individuals from getting risk-averse recourse policies if we limit the target/goal states based on CE methods. This is not surprising since CE methods are not privy to information on uncertainty; this helps emphasize the need to consider an explicit model of recourse actions for risk-averse recourse, rather than use CE methods for recourse.

A.8.4 Disparity in Risks across Gender

We perform *Mann–Whitney U test* [252] to see if the disparity in the mean value of risk measures across the two gender groups in AID and GCD datasets is statistically significant. We present the p-values for significance in the for cost-mean μ_{cost} and cost-variance σ^2_{cost} in Table in Table A.10. We first measure cost-mean and cost-variance for each instance in the entire dataset and then group the measures into two gender groups for the Mann-Whitney U test. We do the same tests for VaR$_{80}$ and CVaR$_{80}$ in Table A.11, and for VaR$_{95}$ and CVaR$_{95}$ in Table A.12. There is a clear statistically significant difference between gender groups for AID, but not for GCD due to limited data points available in the dataset.

A.8.5 Parametric Analysis: Using a Different Recourse Model of Costs R and Transitions T

We conduct an additional experiment on risk measures and disparity in risks across gender groups using a different set of action costs and transitions—these follow the same qualitative assumptions as previously discussed—and report the results in Table A.6. This set of models presumes similar qualitative relationships as Table A.5 with adjusted cost magnitude and action success likelihoods. The results for this different recourse model are summarized in Table A.13. From this set of results, G-RSVI can still compute risk-averse policies that have lower variance in cost σ^2_{cost}, lower VaR, and lower CVaR, compared to risk-neutral policies. Also, we observe similar gender disparity as with the first recourse action model—the risk measures are in favor of the male group, except for Var_{95} and CVar_{95} in GCD. This could be because when the confidence level (α) is high, the tail-end distribution of costs across the two gender groups is similar.

Adult Income		
Action Space	Action Costs $r(a)$	Transitions (Prob. of Success) $p(s')$
Impr-Edu	2.0	HS:0.9, Bachelors:0.8, Masters:0.9, Doc. 0.7
Incr-Hrs	0.5	1.0
Work-Gov	1.2	0.6
Work-Self	2.0	0.9
Work-Private	1.0	0.8
Job-Prof	1.0	0.4
Job-WhiteCol	1.0	0.7
German Credit		
Impr-Skill	2.0	Skilled:0.7, Manage:0.7
Incr-Savings	1.0	0.9
Incr-Checkings	1.0	0.9
Incr-Duration	2.0	1.0
House-Free	2.0	0.9
House-Rent	1.5	0.6
House-Own	1.0	0.3

Table A.6: Action Cost and Transition Models; If the action fails, the state stays the same.

A.8.6 Parametric Analysis: Risk-Aversion β and Horizon H

We vary two parameters used in G-RSVI, risk-aversion level $\beta = 0, 0.25, 0.5, 1.0, 2.0$ and horizon $H = 12, 8, 4$, and evaluate the performance on each of the parameter combinations in Table A.14. We use the full GCD dataset and 2000 randomly selected instances $n = 2000$ in AID for evaluation. We also report the time spent for computing the policies.[3] The measures again indicate better risk measures in risk-averse policies albeit with higher mean cost. The success rate significantly drops when the risk-aversion level $\beta = 2.0$ and when horizon $H = 4$ for AID (shaded in red). This happens because the emphasis on cost-variance is so strong, that it dwarfs the effect of the mean-cost, which in turn prevents the policy from getting to the favorable goal states within the finite horizon. The success rate and other risk measures for *each* datapoint are evaluated using data from 100 Monte Carlo rollouts by following a policy. The ρ_H estimates the success rate in the 100 rollouts. Empirically, ρ_H is expected to be < 1.0 with the presence of uncertainties.

A.8.7 Health Insurance Premiums Dataset

We also experimented on a Health Insurance Premiums Dataset (HIPD) from Kaggle [253]. The dataset overview after pre-processing is shown in Table A.7. The cost and transition models are included in Table A.8. The evaluation in Table A.15 also shows better risk measures for the risk-averse policies computed, compared to the risk-neutral policy. The success rate is low because only three features are actionable: exercising, quitting smoking, and moving to a different region.

A.8.8 Lower Partial Standard Deviation

G-RSVI penalizes the standard deviation σ of the state values as stated in Equation A.3. Alternatively, we can penalize the Lower Partial Standard Deviation [243] which helps avoid situations requiring higher-than-average costs. In this case, the action value becomes:

$$Q_h(s, a) = \mu - \beta\sigma_{LP}[r(\cdot) + V_{h+1}(s')],$$

[3]The experiment is conducted on a MacOS with Apple M1 Max chip and RAM 32GB.

HIPD Feature Values Target: Charges={0,1} #States: 3456 Accuracy: 0.90 (Rand.Forest)	
BMI	<20, <25, <30, <35, <40, Above
Smoker	Yes, No
Region	SW, SE, NW, NE
Children#	0, 1, 2, 3, 4
Age	Young, Twenties, Twenties, Thirties, Forties, Fifties, Older
Gender	Female, Male

Table A.7: The feature list and discrete values for HIPD. Bolded features are immutable. The other features are mutable and actionable.

Health Insurance Premiums		
Action Space	Action Costs $r(a)$	Transitions (Prob. of Success) $p(s')$
Exercise	0.8	0.95
QuitSmoke	0.8	0.5
Move-SW	1.0	0.7
Move-SE	1.0	0.7
Move-NW	1.0	0.7
Move-NE	1.0	0.7

Table A.8: Action Cost and Transition Models; If the action fails, the state stays the same.

where $\mu = \mathbb{E}_{s'}[r(\cdot) + V_{h+1}(s')]$ is the average value, and

$$\sigma_{LP} = \sqrt{\sum_{s'} p(s')[r(\cdot) + V_{h+1}(s') - \mu]^2},$$

$$\forall s' : r(\cdot) + V_{h+1}(s') < \mu.$$

The evaluation on AID and GCD are included in Table A.16, similar trends are observed as the results from Table A.1.

A.8.9 Synthetic Motivation Example

We consider a synthetic motivational example for reducing health insurance premiums. This motivational example is different from the dataset in Section A.8.7; this example uses a simpler set of states and possible actions to illustrate difference policies and risk. For this example we define a model of actions (transition probabilities and costs) as well as a decision function f that a

Dataset	Policy	$\rho_{H=12}$	$(\mu_{cost}, \sigma^2_{cost})$	VaR_{80}	CVaR_{80}	VaR_{95}	CVaR_{95}	Spars.	Proxi.
Adult Income ($n = 100$) (Age < 30)	$\text{DiCE10}_{\beta=0}$	0.774	(5.70, 1.78)	5.14	7.48	6.21	8.64	**2.42**	4.94
	$\beta = 0$	0.997	(**4.89**, 1.33)	5.19	7.15	**6.00**	**8.10**	2.75	**4.37**
	$\beta = 0.25$	0.998	(4.91, 1.08)	5.06	**7.07**	6.03	8.18	2.80	4.56
	$\beta = 0.5$	0.995	(4.95, **1.01**)	**5.04**	7.11	6.05	8.15	2.85	4.68
German Credit ($n = 100$)	$\text{DiCE10}_{\beta=0}$	0.980	(2.22, 0.78)	2.71	4.38	3.67	5.58	**1.39**	**1.63**
	$\beta = 0$	1.000	(**1.53**, 0.34)	1.78	3.56	2.29	**4.29**	**1.21**	**1.27**
	$\beta = 0.25$	1.000	(1.55, 0.29)	1.73	3.51	2.29	4.39	1.23	1.32
	$\beta = 0.5$	1.000	(1.56, **0.25**)	**1.72**	**3.47**	**2.23**	4.30	1.30	1.39

Table A.9: Compare risk-aware policies to the policy computed by using the final CEs provided by DiCE; We randomly select $n = 100$ individual instances from AID and GCD; the same evaluation procedures are followed as Table A.1.

Dataset	Policy	Female μ_{cost}	Male μ_{cost}	p-value	Female σ^2_{cost}	Male σ^2_{cost}	p-value
Adult Income	$\beta = 0$	4.56	2.84	<0.0001	1.41	1.12	<0.0001
(Female, $n = 9824$)	$\beta = 0.25$	4.57	2.87	<0.0001	1.18	0.72	<0.0001
(Male, $n = 16099$)	$\beta = 0.5$	4.61	2.98	<0.0001	1.11	0.57	<0.0001
German Credit	$\beta = 0$	1.72	1.61	0.3531	0.56	0.43	0.1688
(Female, $n = 103$)	$\beta = 0.25$	1.75	1.62	0.2783	0.38	0.31	0.0964
(Male, $n = 178$)	$\beta = 0.5$	1.79	1.65	0.3346	0.33	0.29	0.1637

Table A.10: Mann-Whitney U test results for cost-mean and cost-variance between gender groups for AID and GCD

heath insurance company uses to determine the insurance premium. In this example, consider that insurance premiums are charged at two levels (high, and low). Let us consider an individual with the following features $s_o = [\text{smoking, drinking, high cholesterol, high BMI, west}]$ is classified to receive the high insurance premium, which is unfavorable $f(s_o) = y^-$. Figure A.4a describes the possible actions the individual might take at each state with the transition (success) probabilities , where $T \in \{T1, T2, T3\}$ are the states with the desired outcomes $f(T) = y^+$ (low insurance premium). We assume the cost is the same per step (time cost of 3-months). Taking the *quit-drinking* action at s_o would lead to $T1$ directly (red path) but only with 50% success rate in three months; taking the *quit-smoking* and *move-to-midwest* actions would follow the blue path, where each step has success rate 90%; the *health-diet* and *exercise* actions would progress the feature state along the green path to reduce cholesterol and BMI with 100% success rate but the policy would take 9 months. The feature state remains the same when the action fails.

We apply the G-RSVI algorithm with risk-aversion levels $\beta = 0, 0.5,$ and 1 for risk-neutral, medium risk-averse, and high risk-averse policies respectively. The possible encountered state se-

Dataset	Policy	Female VaR_{80}	Male VaR_{80}	p-value	Female CVaR_{80}	Male CVaR_{80}	p-value
Adult Income	$\beta = 0$	4.76	3.27	<0.0001	7.07	5.79	<0.0001
(Female, $n = 9824$)	$\beta = 0.25$	4.64	3.08	<0.0001	6.95	5.51	<0.0001
(Male, $n = 16099$)	$\beta = 0.5$	4.66	3.10	<0.0001	6.99	5.48	<0.0001
German Credit	$\beta = 0$	2.08	1.89	0.1678	3.70	3.64	0.9289
(Female, $n = 103$)	$\beta = 0.25$	2.00	1.81	0.1934	3.61	3.50	0.5978
(Male, $n = 178$)	$\beta = 0.5$	2.02	1.84	0.1677	3.64	3.50	0.6674

Table A.11: Mann-Whitney U test results for VaR_{80} and CVaR_{80} between gender groups for AID and GCD

Dataset	Policy	Female VaR_{95}	Male VaR_{95}	p-value	Female CVaR_{95}	Male CVaR_{95}	p-value
Adult Income	$\beta = 0$	5.70	4.30	<0.0001	8.00	7.26	<0.0001
(Female, $n = 9824$)	$\beta = 0.25$	5.48	3.95	<0.0001	7.84	7.15	<0.0001
(Male, $n = 16099$)	$\beta = 0.5$	5.16	3.92	<0.0001	7.89	7.08	<0.0001
German Credit	$\beta = 0$	2.84	2.61	0.3029	4.66	4.62	0.7116
(Female, $n = 103$)	$\beta = 0.25$	2.67	2.41	0.1353	4.63	4.42	0.3763
(Male, $n = 178$)	$\beta = 0.5$	2.59	2.33	0.1814	4.51	4.29	0.5426

Table A.12: Mann-Whitney U test results for VaR_{95} and CVaR_{95} between gender groups for AID and GCD

quences (traces) for three polices are shown in Figure A.4b, and the lowest cost trace is highlighted with line thickness indicating the probability. Using 1000 rollouts of each policy, the risk measures of the three policies are summarized in Table A.17. The success rate $\rho_{H=8}$ –which is success probability of recourse within 8 horizon steps– can be less than 100% because some actions have less than 100% probability of success. When $\beta > 0$, the recourse policy achieves lower σ^2_{cost} and lower cost in VaR_{95} and CVaR_{95}, in which the individual has high confidence of a successful recourse with low cost-risk. CVaR_{95} for $\beta = 1$ is n/a because, 100% of the time, recourse is achieved within cost 3. By changing the risk-aversion parameter β, we are able to provide a set of policies corresponding to different risk tolerance levels for the individual.

A.8.10 Methodology

A.8.10.1 Greedy Risk-Sensitive Episodic Value Iteration (G-RSEVI)

G-RSVI is suitable for fast policy computation for all possible recourse states. However, recourse is often tailored and offered to a single individual, and the majority of the states are not reachable by the individual depending on their initial state or current features. Therefore, instead

Dataset	Policy	$\rho_{H=12}$	$(\mu_{cost}, \sigma^2_{cost})$	VaR_{80}	CVaR_{80}	VaR_{95}	CVaR_{95}	Spars.	Proxi.
Adult Income (Female, $n = 1000$)	$\beta = 0$	0.989	(**4.51**, 1.71)	4.75	7.01	5.53	7.86	**2.57**	**3.82**
	$\beta = 0.25$	0.986	(**4.51**, 1.61)	**4.71**	**7.00**	**5.45**	**7.83**	2.59	3.85
	$\beta = 0.5$	0.977	(4.56, **1.43**)	4.81	7.07	5.46	**7.83**	2.61	3.93
(Male, $n = 1000$)	$\beta = 0$	0.997	(**2.67**, 0.96)	2.97	**5.58**	3.86	**6.69**	**1.95**	**2.51**
	$\beta = 0.25$	0.998	(2.71, 0.81)	**2.96**	**5.58**	3.86	6.74	1.97	2.63
	$\beta = 0.5$	0.999	(2.79, **0.69**)	2.98	5.65	**3.78**	6.81	2.05	2.73
German Credit (Female, $n = 103$)	$\beta = 0$	0.999	(**1.65**, 0.32)	1.69	2.85	**2.49**	**3.84**	**1.22**	**1.38**
	$\beta = 0.25$	1.000	(1.67, 0.26)	1.68	2.84	2.59	4.03	1.24	1.41
	$\beta = 0.5$	1.000	(1.67, **0.24**)	**1.66**	**2.80**	2.57	4.00	1.24	1.42
(Male, $n = 178$)	$\beta = 0$	1.000	(**1.62**, 0.26)	1.64	II2.79	2.51	3.87	**1.20**	**1.36**
	$\beta = 0.25$	1.000	(1.63, 0.23)	1.62	2.77	2.46	3.79	1.21	1.38
	$\beta = 0.5$	1.000	(1.63, **0.22**)	**1.61**	**2.74**	**2.42**	**3.62**	1.22	1.39

Table A.13: Evaluating recourse policies across gender using a different set of cost and transition functions; the same evaluation procedures are followed as Table A.1.

of exploring the full state space, we introduce G-RSEVI in Algorithm 7, an episodic variant of G-RSVI that finds the policy for a certain initial state. We allow E episodes for computing the policy (Line 2), and the recourse always starts at the initial state of the recipient (Line 4). For each episode, we select the action based on the current action value $Q_h(s, a)$ with ϵ-greedy, and ϵ is decaying exponentially. We then collect the state-action pairs for H steps (Line 5-8). With the stored transition data D, we update the action value $Q_h(s, a)$ and state value $V_h(s)$ with the risk-averse objective backward (Line 10-17). G-RSEVI ($\mathcal{O}(|E||S||A||H|)$) provides additional benefits in computation complexity when the state space is extremely high (i.e., when $|E| \ll |S|$), compared to G-RSVI ($\mathcal{O}(|S^2||A||H|)$).

We visualize the total recourse cost vs. episode in Figure A.5 for the two example instances selected from AID and GCD (same instances as in Table A.1). Towards the last 500 episodes (low chance of random actions), we show the variability in costs for higher risk-aversion policies is less than the risk-neutral ones.

A.8.10.2 Case Study: Example Instances in Datasets

In this section, we provide additional visualizations of policies computed by G-RSVI as shown in Figure A.6. Instance #16 in AID and Instance #44 in GCD correspond to the two example instances used in Table A.1 as well as in Figure A.5.

Dataset	Policy	ρ_H	$(\mu_{cost}, \sigma^2_{cost})$	VaR_{80}	CVaR_{80}	VaR_{95}	CVaR_{95}	Spars.	Proxi.
Adult Income	$\beta = 0$	0.995	(**3.69**, 1.26)	3.97	6.46	4.87	7.57	**2.20**	**3.05**
($n = 2000$)	$\beta = 0.25$	0.995	(3.71, 0.93)	3.79	6.26	4.63	7.49	2.25	3.23
Horizon $= 12$	$\beta = 0.5$	0.994	(3.78, 0.83)	3.82	6.28	4.62	7.49	2.29	3.34
Time: 23.12 (s)	$\beta = 1.0$	0.978	(4.29, 0.75)	4.12	6.96	4.69	8.05	2.38	3.78
	$\beta = 2.0$	0.717	(3.82, **0.28**)	**3.34**	**5.72**	**4.06**	**7.24**	2.28	3.36
	$\beta = 0$	0.915	(3.30, 0.83)	3.77	5.85	4.80	6.74	**2.14**	**2.77**
	$\beta = 0.25$	0.905	(3.29, 0.60)	3.57	5.63	4.55	6.71	2.21	2.89
Horizon $= 8$	$\beta = 0.5$	0.891	(3.28, 0.56)	3.52	5.60	4.51	6.74	2.24	2.90
Time: 15.89 (s)	$\beta = 1.0$	0.852	(3.48, 0.43)	3.60	5.75	4.52	6.89	2.33	3.13
	$\beta = 2.0$	0.611	(**3.13**, **0.16**)	**3.10**	**5.15**	**3.73**	**5.15**	2.19	2.86
	$\beta = 0$	0.544	(**1.97**, 0.19)	2.22	3.14	2.54	3.74	**1.69**	**1.79**
	$\beta = 0.25$	0.548	(2.00, 0.15)	2.20	**3.10**	2.50	**3.55**	1.76	1.87
Horizon $= 4$	$\beta = 0.5$	0.552	(2.05, 0.12)	2.19	3.22	2.51	3.77	1.83	1.95
Time: 7.90 (s)	$\beta = 1.0$	0.517	(2.09, 0.09)	2.19	3.39	2.47	3.78	1.88	2.02
	$\beta = 2.0$	0.335	(2.13, **0.03**)	**2.12**	3.27	**2.35**	3.96	1.73	2.08
German Credit	$\beta = 0$	1.000	(**1.65**, 0.48)	1.96	3.66	2.63	4.56	**1.26**	**1.33**
($n = 281$)	$\beta = 0.25$	1.000	(1.67, 0.35)	**1.87**	**3.51**	2.51	4.50	1.34	1.43
Horizon$= 12$	$\beta = 0.5$	1.000	(1.70, 0.30)	1.90	3.56	2.48	**4.40**	1.40	1.50
Time: 60.02 (s)	$\beta = 1.0$	1.000	(1.93, **0.25**)	2.04	3.90	2.57	4.74	1.62	1.79
	$\beta = 2.0$	0.981	(2.88, 0.65)	2.03	4.44	**2.38**	4.53	1.82	2.08
	$\beta = 0$	0.998	(**1.64**, 0.43)	1.94	3.58	2.65	4.45	**1.26**	**1.33**
	$\beta = 0.25$	0.999	(1.66, 0.33)	**1.88**	**3.51**	2.52	4.47	1.34	1.42
Horizon$= 8$	$\beta = 0.5$	1.000	(1.70, 0.29)	1.92	3.56	**2.50**	**4.41**	1.40	1.50
Time: 40.45 (s)	$\beta = 1.0$	1.000	(1.91, **0.23**)	2.03	3.90	2.55	4.69	1.61	1.79
	$\beta = 2.0$	0.983	(2.49, 0.31)	2.44	4.75	2.92	5.61	1.81	2.08
	$\beta = 0$	0.963	(**1.54**, 0.22)	1.81	**3.05**	2.37	3.42	**1.26**	**1.32**
	$\beta = 0.25$	0.969	(1.56, 0.19)	**1.78**	3.06	**2.24**	3.40	1.31	1.37
Horizon$= 4$	$\beta = 0.5$	0.972	(1.59, 0.18)	1.81	3.06	2.25	**3.37**	1.35	1.42
Time: 20.56 (s)	$\beta = 1.0$	0.975	(1.69, 0.16)	1.86	3.24	2.25	3.75	1.51	1.59
	$\beta = 2.0$	0.925	(2.18, **0.07**)	2.24	3.68	2.51	3.91	1.71	1.99

Table A.14: Parametric evaluation on risk-aversion β and horizon H for AID and GCD; the same evaluation procedures are followed as Table A.1. The time spent to compute the policies for all states (57600 in AID, 147456 in GCD) is included.

Dataset	Policy	$\rho_{H=12}$	$(\mu_{cost}, \sigma^2_{cost})$	VaR_{80}	CVaR_{80}	VaR_{95}	CVaR_{95}	Spars.	Proxi.
	$\beta = 0$	0.717	(1.74, 0.86)	2.29	3.62	3.42	4.84	1.21	**1.38**
HIPD	$\beta = 0.25$	0.717	(**1.73**, 0.85)	**2.28**	3.62	3.41	4.82	1.21	**1.38**
($n = 604$)	$\beta = 0.5$	0.717	(1.75, **0.82**)	2.29	**3.59**	**3.36**	**4.80**	**1.20**	1.43
	$\beta = 1.0$	0.716	(2.79, 0.83)	3.29	4.58	4.42	5.88	1.66	2.68

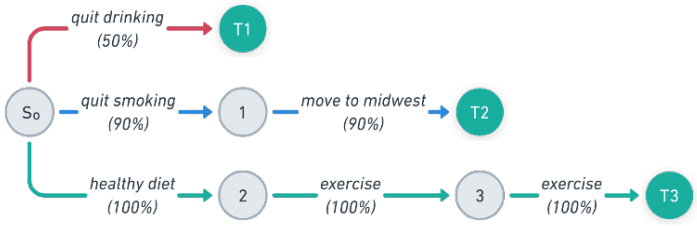
Table A.15: Evaluating recourse policies for Health Insurance Premiums Dataset; the same evaluation procedures are followed as Table A.1.

Dataset	Policy	$\rho_{H=12}$	$(\mu_{cost}, \sigma^2_{cost})$	VaR_{80}	CVaR_{80}	VaR_{95}	CVaR_{95}	Spars.	Proxi.
	$\beta = 0$	0.994	(**3.49**, 1.23)	3.81	6.31	4.76	7.53	**2.09**	**2.87**
Adult Income	$\beta = 0.25$	0.994	(3.51, 0.91)	**3.65**	**6.10**	4.46	7.44	2.17	3.05
($n = 25923$)	$\beta = 0.5$	0.994	(3.53, 0.86)	**3.65**	6.12	**4.44**	**7.43**	2.18	3.09
	$\beta = 0.75$	0.992	(3.60, **0.76**)	3.67	6.13	4.45	**7.43**	2.21	3.20
	$\beta = 0$	1.000	(**1.65**, 0.48)	1.96	3.66	2.63	4.56	**1.26**	**1.33**
German Credit	$\beta = 0.25$	1.000	(1.66, 0.38)	**1.88**	**3.56**	2.53	4.52	1.33	1.41
($n = 281$)	$\beta = 0.5$	1.000	(1.68, 0.33)	1.91	3.56	2.50	**4.46**	1.36	1.45
	$\beta = 0.75$	1.000	(1.74, **0.28**)	1.93	3.72	**2.47**	4.73	1.50	1.59

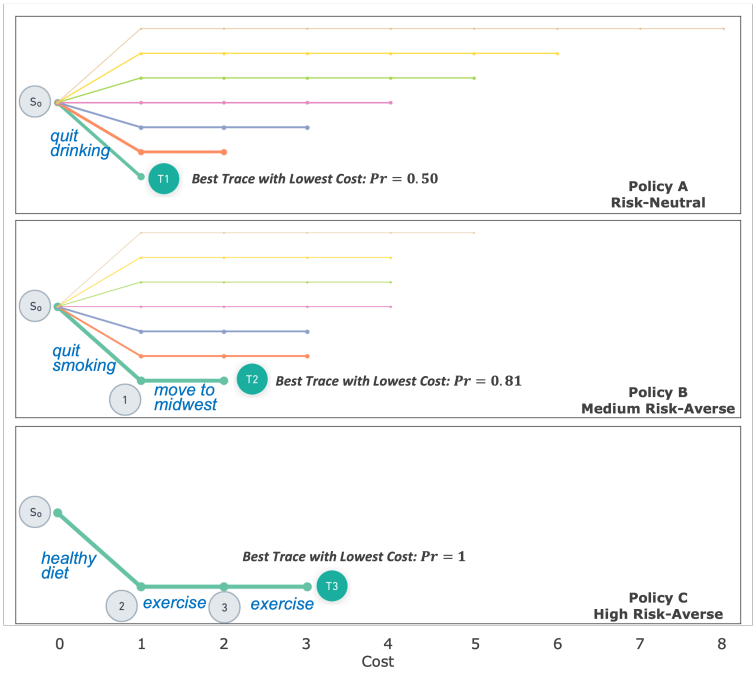
Table A.16: Evaluating recourse policies by penalizing LPSD with different risk-aversion levels β and horizon $H = 12$ for AID, GCD; the same evaluation procedures are followed as Table A.1.

Policy	$\rho_{H=8}$	$(\mu_{cost}, \sigma^2_{cost})$	VaR_{95}	CVaR_{95}
$\beta = 0$	0.999	(**1.9**, 1.7)	5.0	6.5
$\beta = 0.5$	1.000	(2.2, 0.2)	3.0	4.1
$\beta = 1.0$	1.000	(3.0, **0.0**)	**3.0**	**n/a**

Table A.17: Evaluating risk measures



(a) Transition probabilities



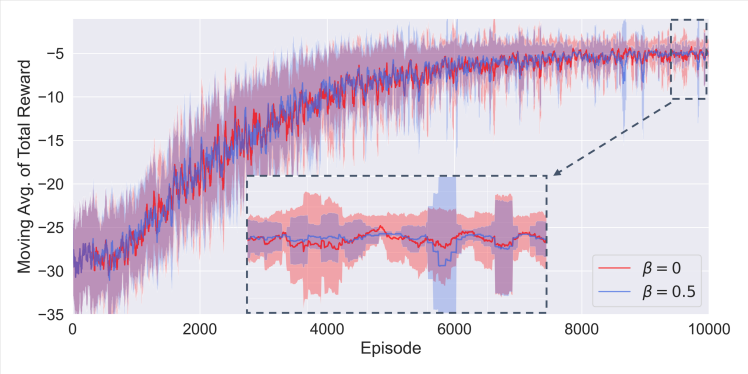
(b) Recourse policy visualization

Figure A.4: Health insurance premiums recourse (synthetic)

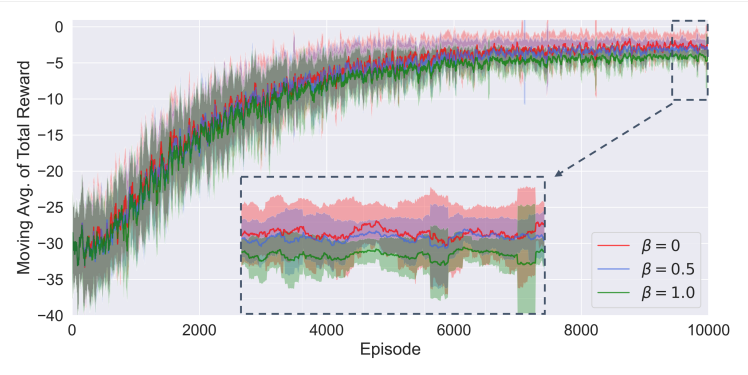
Algorithm 7 G-RSEVI

Input: recourse MDP$\langle S, A, T, R, H\rangle$, ML model f, initial feature state s_o
Parameters: risk aversion level $\beta \in [0, \infty)$, max episode E

$V_{H+1}(s) \leftarrow 0, \forall s \in S$
for episode $k = 1, \ldots, E$ **do**
  $D \leftarrow empty$
  $s_1 \leftarrow$ Reset state to s_o
  for step $h = 1, \ldots, H$ **do**
    $a_h \leftarrow \operatorname{argmax} Q_h(s_h, \cdot)$ ϵ-greedy action selection
    $D \leftarrow$ store (s_h, a_h)
    $s_{h+1} \leftarrow$ take action and receive the next state
  end for
  for step $h = H, H-1, \ldots, 1$ **do**
    $s, a \leftarrow$ get (s_h, a_h) from D
    $r(s'), p(s') \leftarrow$ receive reward and transition
    $\mu \leftarrow \sum_{s'} p(s')[r(s') + V_{h+1}(s')]$
    $\sigma^2 \leftarrow \sum_{s'} p(s')[r(s') + V_{h+1}(s') - \mu]^2$
    $Q_h(s, a) \leftarrow \mu - \beta\sigma$ (Equation A.3)
    $V_h(s) \leftarrow \max Q_h(s, \cdot)$
    $\pi_h(s) = \operatorname{argmax} Q_h(s, \cdot)$
  end for
end for
return Recourse policy $\pi_h(s)$

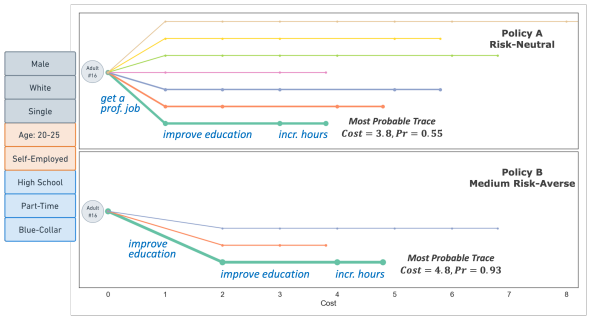


(a) Example instance in AID

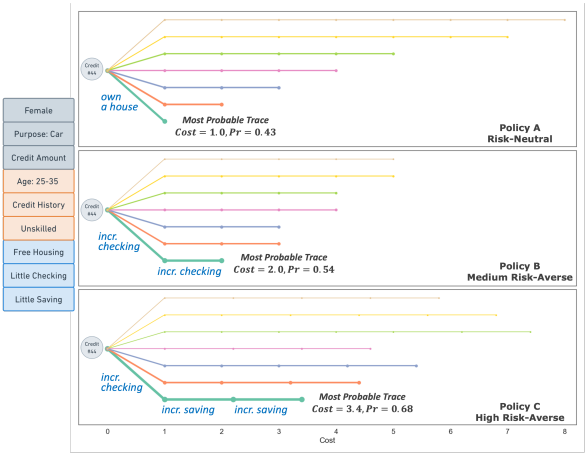


(b) Example instance in GCD

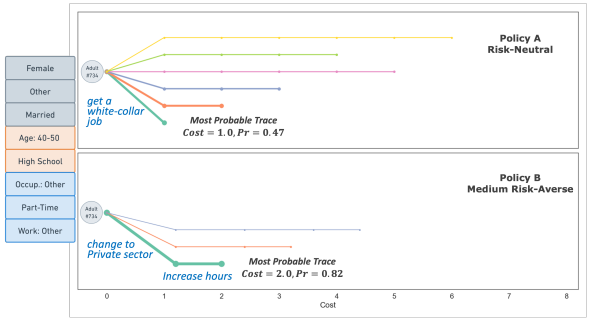
Figure A.5: Moving average of total recourse cost using G-RSEVI with maximum episode 10,000 and exponential decay factor 0.9995 for ϵ. Shades indicate $\pm\sigma$ cost range of the moving window.



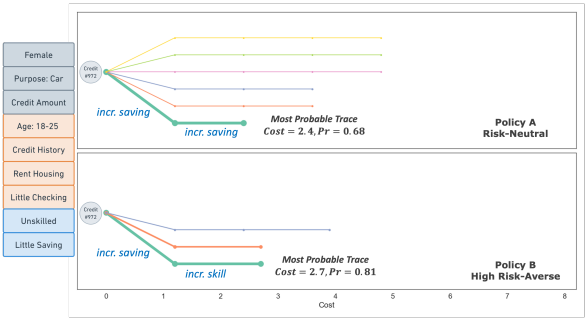
(a) Instance #16 in AID



(b) Instance #44 in GCD



(c) Instance #734 in AID



(d) Instance #972 in GCD

Figure A.6: Policy visualizations; $\beta = 0, 0.5, 1.0$ correspond to risk-neutral, medium risk-averse, and high-risk-averse respectively.

BIBLIOGRAPHY

[1] Ayman M. A. Youssef and Hoda A. ElMaraghy. Optimal configuration selection for reconfigurable manufacturing systems. *International Journal of Flexible Manufacturing Systems*, 19(2):67–106, Jun 2007.

[2] Sarvapali D. Ramchurn, Feng Wu, Wenchao Jiang, Joel E. Fischer, Steve Reece, Stephen Roberts, Tom Rodden, Chris Greenhalgh, and Nicholas R. Jennings. Human—agent collaboration for disaster response. *Autonomous Agents and Multi-Agent Systems*, 30(1):82–111, jan 2016.

[3] Jeffrey L Adler and Victor J Blue. A cooperative multi-agent transportation management and route guidance system. *Transportation Research Part C: Emerging Technologies*, 10(5):433–454, 2002.

[4] Baicun Wang, Pai Zheng, Yue Yin, Albert Shih, and Lihui Wang. Toward human-centric smart manufacturing: A human-cyber-physical systems (hcps) perspective. *Journal of Manufacturing Systems*, 63:471–490, 2022.

[5] Gokce B. Laleci, Asuman Dogac, Mehmet Olduz, Ibrahim Tasyurt, Mustafa Yuksel, and Alper Okcan. Saphire: A multi-agent system for remote healthcare monitoring through computerized clinical guidelines. *Agent Technology and e-Health*, pages 25–44, 2008.

[6] Rodrigo Bavaresco, Diórgenes Silveira, Eduardo Reis, Jorge Barbosa, Rodrigo Righi, Cristiano Costa, Rodolfo Antunes, Marcio Gomes, Clauter Gatti, Mariangela Vanzin, Saint Clair Junior, Elton Silva, and Carlos Moreira. Conversational agents in business: A systematic literature review and future research directions. *Computer Science Review*, 36:100239, 2020.

[7] Thomas O'Neill, Nathan McNeese, Amy Barron, and Beau Schelble. Human–autonomy teaming: A review and analysis of the empirical literature. *Human Factors*, 64(5):904–938, 2022. PMID: 33092417.

[8] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[9] Seenu N., Kuppan Chetty R.M., Ramya M.M., and Mukund Nilakantan Janardhanan. Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems. *Industrial Robot: the international journal of robotics research and application*, 47(6):929–942, January 2020.

[10] R. Nair, M. Tambe, M. Yokoo, Pynadath D., and S. Marsella. Taming decentralized pomdps: towards efficient policy computation for multiagent settings. *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 705–711, 2013.

[11] S. Amador, Steven Okamoto, and R. Zivan. Dynamic multi-agent task allocation with spatial and temporal constraints. *International Conference on Autonomous Agents and Multiagent Systems*, 28(1):1495–1496, 2014.

[12] Bo Fu, William Smith, Denise M. Rizzo, Matthew Castanier, Maani Ghaffari, and Kira Barton. Robust task scheduling for heterogeneous robot teams under capability uncertainty. *IEEE Transactions on Robotics*, 39(2):1087–1105, 2023.

[13] Gennaro Notomista, Siddharth Mayya, Seth Hutchinson, and Magnus Egerstedt. An optimal task allocation strategy for heterogeneous multi-robot systems. *2019 18th European Control Conference (ECC)*, pages 2071–2076, 2019.

[14] Han-Lim Choi, Andrew K. Whitten, and Jonathan P. How. Decentralized task allocation for heterogeneous teams with cooperation constraints. *Proceedings of the 2010 American Control Conference*, pages 3057–3062, 2010.

[15] Yousef Emam, Siddharth Mayya, Gennaro Notomista, Addison Bohannon, and Magnus Egerstedt. Adaptive task allocation for heterogeneous multi-robot teams with evolving and unknown robot capabilities. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7719–7725, 2020.

[16] Bo Fu, William Smith, Denise Rizzo, Matthew Castanier, and Kira Barton. Heterogeneous vehicle routing and teaming with gaussian distributed energy uncertainty. *International Conference on Intelligent Robots and Systems*, 2020.

[17] Jacopo Banfi, Andrew Messing, Christopher Kroninger, Ethan Stump, Seth Hutchinson, and Nicholas Roy. Hierarchical planning for heterogeneous multi-robot routing problems via learned subteam performance. *IEEE Robotics and Automation Letters*, 7(2):4464–4471, 2022.

[18] Xinzhu Liu, Di Guo, Xinyu Zhang, and Huaping Liu. Heterogeneous embodied multi-agent collaboration. *IEEE Robotics and Automation Letters*, 9(6):5377–5384, 2024.

[19] Ruchir Patel, Eliot Rudnick-Cohen, Shapour Azarm, Michael Otte, Huan Xu, and Jeffrey W. Herrmann. Decentralized task allocation in multi-agent systems using a decentralized genetic algorithm. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3770–3776, 2020.

[20] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Uncertainty in Artificial Intelligence*, 27(4), 2002.

[21] Dhouha Ben Noureddine, A. Gharbi, and S. Ahmed. Multi-agent deep reinforcement learning for task allocation in dynamic environment. *International Conference on Software Technologies*, 2017.

[22] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, page 2681–2690, 2017.

[23] Navid Hooshangi and Ali Asghar Alesheikh. Agent-based task allocation under uncertainties in disaster environments: An approach to interval uncertainty. *International Journal of Disaster Risk Reduction*, 24:160–171, 2017.

[24] J. Y. C. Chen and M. J. Barnes. Human–agent teaming for multirobot control: A review of human factors issues. *Transactions on Human-Machine Systems*, 44(1):13–29, 2014.

[25] Mehdi Dadvar, Saeed Moazami, Harley R. Myler, and Hassan Zargarzadeh. Multi-agent task allocation in complementary teams: A hunter and gatherer approach. *CoRR*, abs/1912.05748:1–15, 2019.

[26] Tamzidul Mina, S. S. Kannan, Wonse Jo, and B. Min. Adaptive workload allocation for multi-human multi-robot teams for independent and homogeneous tasks. *IEEE Access*, 8:152697–152712, 2020.

[27] Haochen Wu, Amin Ghadami, Alparslan Emrah Bayrak, Jonathon M. Smereka, and Bogdan I. Epureanu. Impact of heterogeneity and risk aversion on task allocation in multi-agent teams. *IEEE Robotics and Automation Letters*, 6(4):7065–7072, 2021.

[28] Peter Karkus, David Hsu, and Wee Lee. Qmdp-net: Deep learning for planning under partial observability. *International Conference on Neural Information Processing Systems*, page 4697–4707, 2017.

[29] Eric Schneider, Elizabeth Sklar, Simon Parsons, and A. Ozgelen. Auction-based task allocation for multi-robot teams in dynamic environments. *Towards Autonomous Robotic Systems*, pages 246–257, 2015.

[30] A. Maoudj, B. Bouzouia, A. Hentout, and R. Toumi. Multi-agent approach for task allocation and scheduling in cooperative heterogeneous multi-robot team: Simulation results. *IEEE 13th International Conference on Industrial Informatics*, pages 179–184, 2015.

[31] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, North Chelmsford, MA, USA, 2012.

[32] Rachel Holladay, Shervin Javdani, Anca Dragan, and Siddhartha Srinivasa. Active comparison based learning incorporating user uncertainty and noise. *RSS Workshop on Model Learning for Human-Robot Communication*, 2016.

[33] Chandrayee Basu, Erdem Biyik, Zhixun He, Mukesh Singhal, and Dorsa Sadigh. Active learning of reward dynamics from hierarchical queries. *International Conference on Intelligent Robots and Systems*, 2019.

[34] Erika Puiutta and Eric M. S. P. Veith. Explainable reinforcement learning: A survey. *Machine Learning and Knowledge Extraction*, pages 77–95, 2020.

[35] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215, 5 2019.

[36] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding dqns. *Proceedings of The 33rd International Conference on Machine Learning*, 48:1899–1908, 06 2016.

[37] Pedro Sequeira and Melinda Gervasio. Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artificial Intelligence*, 288:103367, 08 2020.

[38] Karl Pearson. Note on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London Series I*, 58:240–242, 1 1895.

[39] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[40] Philip N Sabes and Michael Jordan. Reinforcement learning by probability matching. *Advances in Neural Information Processing Systems*, 8, 1995.

[41] Erin Zaroukian, Sebastian S. Rodriguez, Sean L. Barton, James A. Schaffer, Brandon Perelman, Nicholas R. Waytowich, Blaine Hoffman, and Derrik E. Asher. Algorithmically identifying strategies in multi-agent game-theoretic environments. *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 11006:1100614, 2019.

[42] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438, 1969.

[43] George Sugihara, Robert May, Hao Ye, Chih hao Hsieh, Ethan Deyle, Michael Fogarty, and Stephan Munch. Detecting causality in complex ecosystems. *Science*, 338(6106):496–500, 2012.

[44] Hao Ye, Ethan Deyle, Luis Gilarranz, and George Sugihara. Distinguishing time-delayed causal interactions using convergent cross mapping. *Scientific Reports*, 5:14750, 2015.

[45] Leo Breston, Eric Leonardis, Laleh Quinn, Michael Tolston, Janet Wiles, and Andrea Chiba. Convergent cross sorting for estimating dynamic coupling. *Scientific Reports*, 11:20374, 2021.

[46] James M. McCracken and Robert S. Weigel. Convergent cross-mapping and pairwise asymmetric inference. *Physical Review E*, 90:062903, 2014.

[47] Adam Thomas Clark, Hao Ye, Forest Isbell, Ethan R. Deyle, Jane Cowles, G. David Tilman, and George Sugihara. Spatial convergent cross mapping to detect causal relationships from short time series. *Ecology*, 96(5):1174–1181, 2015.

[48] Yang Zhang, Qingyu Yang, Dou An, and Chengwei Zhang. Coordination between individual agents in multi-agent reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11387–11394, May 2021.

[49] Sean Barton, Nicholas Waytowich, Erin Zaroukian, and Derrik Asher. Measuring collaborative emergent behavior in multi-agent reinforcement learning. *Proceedings of the 1st International Conference on Human Systems Engineering and Design*, 25(3):422–427, 2018.

[50] Elhadji Amadou Oury Diallo and Toshiharu Sugawara. Learning strategic group formation for coordinated behavior in adversarial multi-agent with double dqn. *PRIMA 2018: Principles and Practice of Multi-Agent Systems*, pages 458–466, 2018.

[51] Yuki Miyashita and Toshiharu Sugawara. Analysis of coordinated behavior structures with multi-agent deep reinforcement learning. *Applied Intelligence*, 51(2):1069–1085, 2 2021.

[52] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. *Proceedings of the Seventeenth International Conference on Machine Learning*, page 663–670, 2000.

[53] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[54] Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. *Proceedings of the Fifteenth International Conference on Machine Learning*, page 242–250, 1998.

[55] Justin Fu, Andrea Tacchetti, Julien Perolat, and Yoram Bachrach. Evaluating strategic structures in multi-agent inverse reinforcement learning. *Journal of Artificial Intelligence Research*, 71:925–951, 9 2021.

[56] Xiaomin Lin, Peter A. Beling, and Randy Cogill. Multiagent inverse reinforcement learning for two-person zero-sum games. *IEEE Transactions on Games*, 10(1):56–68, 2018.

[57] Lantao Yu, Jiaming Song, and Stefano Ermon. Multi-agent adversarial inverse reinforcement learning. *Proceedings of the 36th International Conference on Machine Learning*, pages 7194–7201, 2019.

[58] Chelsea Finn, Paul F. Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *CoRR*, abs/1611.03852, 2016.

[59] Andrew Whiten. *Natural Theories of Mind: Evolution, Development and Simulation of Everyday Mindreading*. Blackwell, 1991.

[60] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. *Machine Learning Proceedings 1994*, pages 157–163, 1994.

[61] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021.

[62] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PLOS ONE*, 12(4):1–15, 04 2017.

[63] T. E. Boult, S. Cruz, A.R. Dhamija, M. Gunther, J. Henrydoss, and W.J. Scheirer. Learning and the unknown: Surveying steps toward open world recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9801–9807, Jul. 2019.

[64] Haochen Wu, Pedro Sequeira, and David V. Pynadath. Multiagent inverse reinforcement learning via theory of mind reasoning. *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, page 708–716, 2023.

[65] Sebastian Tschiatschek, Ahana Ghosh, Luis Haug, Rati Devidze, and Adish Singla. Learner-aware teaching: Inverse reinforcement learning with preferences and constraints. *Advances in Neural Information Processing Systems*, 32, 2019.

[66] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Deep inverse reinforcement learning for behavior prediction in autonomous driving: Accurate forecasts of vehicle motion. *IEEE Signal Processing Magazine*, 38(1):87–96, 2021.

[67] Pat Langley. Open-world learning for radically autonomous agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13539–13543, Apr. 2020.

[68] Gyuhak Kim, Changnan Xiao, Tatsuya Konishi, Zixuan Ke, and Bing Liu. A theoretical study on solving continual learning. *Advances in Neural Information Processing Systems*, 35:5065–5079, 2022.

[69] Bing Liu, Sahisnu Mazumder, Eric Robertson, and Scott Grigsby. Ai autonomy: Self-initiated open-world continual learning and adaptation. *AI Magazine*, 44(2):185–199, 2023.

[70] Steven Hansen, Alexander Pritzel, Pablo Sprechmann, Andre Barreto, and Charles Blundell. Fast deep reinforcement learning using online adjustments from the past. *Advances in Neural Information Processing Systems*, 31, 2018.

[71] Jie Xu, Cem Tekin, Simpson Zhang, and Mihaela van der Schaar. Distributed multi-agent online learning based on global feedback. *IEEE Transactions on Signal Processing*, 63(9):2225–2238, 2015.

[72] Georgios Chalkiadakis and Craig Boutilier. Coordination in multiagent reinforcement learning: a bayesian approach. *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, page 709–716, 2003.

[73] Tung Thai, Ming Shen, Neeraj Varshney, Sriram Gopalakrishnan, Utkarsh Soni, Chitta Baral, Jivko Sinapov, and Matthias Scheutz. An architecture for novelty handling in a multi-agent stochastic environment: Case study in open-world monopoly. *Proceedings of AAAI Spring Symposium on Designing Artificial Intelligence for Open Worlds.*, 2022.

[74] Shivam Goel, Yash Shukla, Vasanth Sarathy, Matthias Scheutz, and Jivko Sinapov. Rapid-learn: A framework for learning to recover for handling novelties in open-world environments. *2022 IEEE International Conference on Development and Learning (ICDL)*, pages 15–22, 2022.

[75] Jonathan C. Balloch, Zhiyu Lin, Xiangyu Peng, Mustafa Hussain, Aarun Srinivas, Robert Wright, Julia M. Kim, and Mark O. Riedl. Neuro-symbolic world models for adapting to open world novelty. *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, page 2848–2850, 2023.

[76] R. Cohen, M. G. Lipton, M. Q. Dai, and B. Benhabib. Conceptual Design of a Modular Robot. *Journal of Mechanical Design*, 114(1):117–125, 03 1992.

[77] Yara Rizk, Mariette Awad, and Edward W. Tunstel. Decision making in multiagent systems: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 10(3):514–529, 2018.

[78] Chuangchuang Sun, Shirantha Welikala, and Christos G. Cassandras. Optimal composition of heterogeneous multi-agent teams for coverage problems with performance bound guarantees. *Automatica*, 117:108961, 2020.

[79] Bo Liu, Qiang Liu, Peter Stone, Animesh Garg, Yuke Zhu, and Anima Anandkumar. Coach-player multi-agent reinforcement learning for dynamic team composition. *Proceedings of the 38th International Conference on Machine Learning*, 139:6860–6870, 18–24 Jul 2021.

[80] Joshua Rubini, Robert B. Heckendorn, and Terence Soule. Evolution of team composition in multi-agent systems. *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, page 1067–1074, 2009.

[81] Markus Waibel, Laurent Keller, and Dario Floreano. Genetic team composition and level of selection in the evolution of cooperation. *IEEE Transactions on Evolutionary Computation*, 13(3):648–660, 2009.

[82] Paweł Lichocki, Steffen Wischmann, Laurent Keller, and Dario Floreano. Evolving team compositions by agent swapping. *IEEE Transactions on Evolutionary Computation*, 17(2):282–298, 2013.

[83] Gaganmeet Kaur Awal and K. K. Bharadwaj. Team formation in social networks based on collective intelligence — an evolutionary approach. *Applied Intelligence*, 41(2):627–648, sep 2014.

[84] Mitchell A. Potter and Kenneth A. De Jong. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation*, 8(1):1–29, 03 2000.

[85] Chi-Keong Goh and Kay Chen Tan. A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 13(1):103–127, 2009.

[86] C.K. Goh, K.C. Tan, D.S. Liu, and S.C. Chiam. A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. *European Journal of Operational Research*, 202(1):42–54, 2010.

[87] David Ha. Reinforcement learning for improving agent design. *Artificial Life*, 25(4):352–365, 2019.

[88] Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. *Nature Communications*, 12(1):5721, October 2021.

[89] Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R. Walter. Jointly learning to construct and control agents using deep reinforcement learning. *2019 International Conference on Robotics and Automation (ICRA)*, pages 9798–9805, 2019.

[90] Eddieb Sadat, Mostaan Lotfalian Saremi, and Alparslan Emrah Bayrak. A Two-Timescale Reinforcement Learning Approach for Control Co-Design Problems. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Volume 3A: 49th Design Automation Conference (DAC):V03AT03A003, 08 2023.

[91] Sehoon Ha, Stelian Coros, Alexander Alspach, Joohyung Kim, and Katsu Yamane. Computational co-optimization of design parameters and motion trajectories for robotic systems. *The International Journal of Robotics Research*, 37(13-14):1521–1536, 2018.

[92] Danial Sufiyan, Luke Soe Thura Win, Shane Kyi Hla Win, Gim Song Soh, and Shaohui Foong. Joint mechanical design and flight control optimization of a nature-inspired unmanned aerial vehicle via collaborative co-evolution. *IEEE Robotics and Automation Letters*, 6(2):2044–2051, 2021.

[93] Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. Dynamic team heterogeneity in cooperative coevolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 22(6):934–948, 2018.

[94] Frans A. Oliehoek, Matthijs T. J. Spaan, and Nikos A. Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal Of Artificial Intelligence Research*, 32:289–353, 2011.

[95] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602:1–9, 2013.

[96] E. A. Leicht and M. E. J. Newman. Community structure in directed networks. *Physical Review Letters*, 100:118703, 2008.

[97] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1433–1438, 2008.

[98] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR*, abs/1507.06527:1–7, 2015.

[99] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8, 1992.

[100] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998.

[101] Bingyao Liu, Satinder Singh, Richard Lewis, and Shiyin Qin. Optimal rewards in multiagent teams. *International Conference on Development and Learning and Epigenetic Robotics*, 2012.

[102] Yunlong Dong, Xiuchuan Tang, and Ye Yuan. Principled reward shaping for reinforcement learning via lyapunov stability theory. *Neurocomputing*, 393:83–90, 2020.

[103] Kristina Lerman, Chris Jones, Aram Galstyan, and Maja J Matarić. Analysis of dynamic task allocation in multi-robot systems. *The International Journal of Robotics Research*, 25(3):225–241, 2006.

[104] Maria Gini. Multi-robot allocation of tasks with temporal and ordering constraints. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), 2017.

[105] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. *Innovations in Multi-Agent Systems and Applications - 1*, pages 183–221, 2010.

[106] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, page 50–56, 2016.

[107] Soumya Banerjee and Joshua Hecker. A multi-agent system approach to load-balancing and resource allocation for distributed computing. *Complex Systems Digital Campus 2015 World eConference on Complex Systems*, pages 41–54, 12 2017.

[108] Niall Creech, Natalia Criado Pacheco, and Simon Miles. Resource allocation in dynamic multiagent systems. *CoRR*, abs/2102.08317:1–22, 2021.

[109] Glenn J. Lematta, Craig J. Johnson, Eric Holder, Lixiao Huang, Shawaiz A. Bhatti, and Nancy J. Cooke. Team interaction strategies for human–autonomy teaming in next generation combat vehicles. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 64(1):77–81, 2020.

[110] Ahmed Jamah Ahmed Alnagrat, Rizalafande Che Ismail, and Syed Zulkarnain Syed Idrus. Extended reality (xr) in virtual laboratories: A review of challenges and future training directions. *Journal of Physics: Conference Series*, 1874(1):012031, may 2021.

[111] Michael J. Barnes, Jessie Y.C. Chen, and Susan C. Hill. Humans and autonomy: Implications of shared decision making for military operations. *US Army Research Laboratory*, 2017.

[112] Yannick Brand and Axel Schulte. Workload-adaptive and task-specific support for cockpit crews: design and evaluation of an adaptive associate system. *Human-Intelligent Systems Integration*, 3(2):187–199, Jun 2021.

[113] Shayne Loft, Vanessa Bowden, Janelle Braithwaite, Daniel B. Morrell, Samuel Huf, and Francis T. Durso. Situation awareness measures for simulated submarine track management. *Human Factors*, 57(2):298–310, 2015. PMID: 25850159.

[114] Christian Herff, Dominic Heger, Ole Fortmann, Johannes Hennrich, Felix Putze, and Tanja Schultz. Mental workload during n-back task—quantified in the prefrontal cortex using fnirs. *Frontiers in Human Neuroscience*, 7, 2014.

[115] Yichuan Liu, Hasan Ayaz, and Patricia A. Shewokis. Multisubject "learning" for mental workload classification using concurrent eeg, fnirs, and physiological measures. *Frontiers in Human Neuroscience*, 11, 2017.

[116] Srinidhi Parshi, Rafiul Amin, Hamid Fekri Azgomi, and Rose T. Faghih. Mental workload classification via hierarchical latent dictionary learning: A functional near infrared spectroscopy study. *2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pages 1–4, 2019.

[117] Oscar Martinez Mozos, Virginia Sandulescu, Sally Andrews, David Ellis, Nicola Bellotto, Radu Dobrescu, and Jose Manuel Ferrandez. Stress detection using wearable physiological and sociometric sensors. *International Journal of Neural Systems*, 27(02):1650041, 2017. PMID: 27440466.

[118] Tenzing C. Dolmans, Mannes Poel, Jan-Willem J. R. van 't Klooster, and Bernard P. Veldkamp. Perceived mental workload classification using intermediate fusion multimodal deep learning. *Frontiers in Human Neuroscience*, 14, 2021.

[119] Ruikun Luo, Yifan Wang, Yifan Weng, Victor Paul, Mark J. Brudnak, Paramsothy Jayakumar, Matt Reed, Jeffrey L. Stein, Tulga Ersal, and X. Jessie Yang. Toward real-time assessment of workload: A bayesian inference approach. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 63(1):196–200, 2019.

[120] Siddharth Siddharth and Mohan M. Trivedi. On assessing driver awareness of situational criticalities: Multi-modal bio-sensing and vision-based analysis, evaluations, and insights. *Brain Sciences*, 10(1), 2020.

[121] Ziheng Wang, Ryan M. Hope, Zuoguan Wang, Qiang Ji, and Wayne D. Gray. Cross-subject workload classification with a hierarchical bayes model. *NeuroImage*, 59(1):64–69, 2012. Neuroergonomics: The human brain in action and at work.

[122] Denny Yu, Chantale Wilson Antonik, Frederick Webber, Eric Watz, and Winston Bennett. Correction to: Multi-modal physiological sensing approach for distinguishing high workload events in remotely piloted aircraft simulation. *Human-Intelligent Systems Integration*, 3(2):201–211, Jun 2021.

[123] Clémence Dubois and Jerome Le Ny. Adaptive task allocation in human-machine teams with trust and workload cognitive models. *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3241–3246, 2020.

[124] Keryl A. Cosenzo Raja Parasuraman and Ewart De Visser. Adaptive automation for human supervision of multiple uninhabited vehicles: Effects on change detection, situation awareness, and mental workload. *Military Psychology*, 21(2):270–297, 2009.

[125] Yin Sheng, Haibin Zhu, Xianzhong Zhou, and Wenting Hu. Effective approaches to adaptive collaboration via dynamic role assignment. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(1):76–92, 2016.

[126] Wen-Yu Chao Le Zhang, Qing-Xing Qu and Vincent G. Duffy. Investigating the combination of adaptive uis and adaptable uis for improving usability and user performance of complex uis. *International Journal of Human–Computer Interaction*, 36(1):82–94, 2020.

[127] Gloria Calhoun. Adaptable (not adaptive) automation: Forefront of human–automation teaming. *Human Factors*, 64(2):269–277, 2022. PMID: 34435537.

[128] Haochen Wu, Amin Ghadami, Alparslan Emrah Bayrak, Jonathon M. Smereka, and Bogdan I. Epureanu. Task allocation with load management in multi-agent teams. *2022 International Conference on Robotics and Automation (ICRA)*, pages 8823–8830, 2022.

[129] Christopher G. Lucas, Thomas L. Griffiths, Fei Xu, and Christine Fawcett. A rational model of preference learning and choice prediction by children. *Proceedings of the 21st International Conference on Neural Information Processing Systems*, page 985–992, 2008.

[130] Paul van Gent, Haneen Farah, Nicole van Nes, and Bart van Arem. Analysing noisy driver physiology real-time using off-the-shelf sensors: Heart rate analysis software from the taking the fast lane project. *Journal of Open Research Software*, Oct 2019.

[131] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics*, 9(1):427–438, 2013.

[132] Sven Gronauer and Klaus Dieopold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55:1–49, 2022.

[133] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, page 2586–2591, 2007.

[134] Sriraam Natarajan, Gautam Kunapuli, Kshitij Judah, Prasad Tadepalli, Kristian Kersting, and Jude Shavlik. Multi-agent inverse reinforcement learning. *International Conference on Machine Learning and Applications*, pages 395–400, 2010.

[135] Siqi Liu, Guy Lever, Nicholas Heess, Josh Merel, Saran Tunyasuvunakool, and Thore Graepel. Emergent coordination through competition. *CoRR*, abs/1902.07151:1–19, 2019.

[136] Derrik Asher, Sean Barton, Erin Zaroukian, and Nicholas Waytowich. Effect of cooperative team size on coordination in adaptive multi-agent systems. *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 11006:337–348, 2019.

[137] Carlos Guestrin, Michail G. Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. *Proceedings of the Nineteenth International Conference on Machine Learning*, page 227–234, 2002.

[138] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. *Proceedings of the 20th National Conference on Artificial Intelligence*, 1:133–139, 2005.

[139] Wendelin Boehmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. *Proceedings of the 37th International Conference on Machine Learning*, 119:980–991, 13–18 Jul 2020.

[140] Mark E. J. Newman. Analysis of weighted networks. *Physical Review E*, 70:056131, 2004.

[141] Chih-hao Hsieh, Christian Anderson, and George Sugihara. Extending nonlinear analysis to short ecological time series. *The American Naturalist*, 171(1):71–80, 2008.

[142] Heiko Hamann, Thomas Schmickl, Heinz Wörn, and Karl Crailsheim. Analysis of emergent symmetry breaking in collective decision making. *Neural Computing and Applications*, 21:207–218, 03 2012.

[143] Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, 2022.

[144] Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse: From counterfactual explanations to interventions. *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, page 353–362, 2021.

[145] Julius Von Kügelgen, Amir-Hossein Karimi, Umang Bhatt, Isabel Valera, Adrian Weller, and Bernhard Schölkopf. On the fairness of causal algorithmic recourse. *Proceedings of the AAAI conference on artificial intelligence*, 36(9):9584–9594, 2022.

[146] Sahil Verma, Keegan Hines, and John P. Dickerson. Amortized generation of sequential algorithmic recourses for black-box models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8512–8519, 2022.

[147] Ronald A Howard and James E Matheson. Risk-sensitive markov decision processes. *Management science*, 18(7):356–369, 1972.

[148] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. *Advances in neural information processing systems*, 28, 2015.

[149] Nicole Bäuerle and Ulrich Rieder. More risk-sensitive markov decision processes. *Mathematics of Operations Research*, 39(1):105–120, 2014.

[150] Glyn A Holton. *Value-at-Risk: Theory and Practice, Second Edition*. Online, 2013.

[151] R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at risk. *Journal of Risk*, 2(3):21–41, 2000.

[152] Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

[153] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, page 4950–4957, 2018.

[154] Haochen Wu, Amin Ghadami, Alparslan Emrah Bayrak, Jonathon M. Smereka, and Bogdan I. Epureanu. Evaluating emergent coordination in multi-agent task allocation through causal inference and sub-team identification. *IEEE Robotics and Automation Letters*, 8(2):728–735, 2023.

[155] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[156] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[157] Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. *Advances in Neural Information Processing Systems*, 20, 2007.

[158] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. Maximum margin planning. *Proceedings of the Twenty-Third International Conference on Machine Learning*, page 729–736, 2006.

[159] Tummalapalli Sudhamsh Reddy, Vamsikrishna Gopikrishna, Gergely Zaruba, and Manfred Huber. Inverse reinforcement learning for decentralized non-cooperative multiagent systems. *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1930–1935, 2012.

[160] Xingyu Wang and Diego Klabjan. Competitive multi-agent inverse reinforcement learning with sub-optimal demonstrations. *Proceedings of the Thirty-Fifth International Conference on Machine Learning*, 80:5143–5151, 2018.

[161] Julian Jara-Ettinger. Theory of mind as inverse reinforcement learning. *Current Opinion in Behavioral Sciences*, 29:105–110, 2019. Artificial Intelligence.

[162] David V. Pynadath, Paul S. Rosenbloom, and Stacy C. Marsella. Reinforcement learning for adaptive theory of mind in the sigma cognitive architecture. *Artificial General Intelligence*, pages 143–154, 2014.

[163] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, S. M. Ali Eslami, and Matthew Botvinick. Machine theory of mind. *Proceedings of the 35th International Conference on Machine Learning*, 80:4218–4227, 07 2018.

[164] Sriraam Natarajan, Gautam Kunapuli, Kshitij Judah, Prasad Tadepalli, Kristian Kersting, and Jude Shavlik. Multi-agent inverse reinforcement learning. *2010 Ninth International Conference on Machine Learning and Applications*, pages 395–400, 2010.

[165] David V. Pynadath and Stacy C. Marsella. Psychsim: Modeling theory of mind with decision-theoretic agents. *International Joint Conferences on Artificial Intelligence*, page 1181–1186, 2005.

[166] Christopher Amato and Frans A. Oliehoek. Scalable planning and learning for multiagent pomdps. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, page 1995–2002, 2015.

[167] David V. Pynadath and Milind Tambe. The communicative multiagent team decision problem: analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16(1):389–423, jun 2002.

[168] B. Fuglede and F. Topsoe. Jensen-Shannon divergence and Hilbert space embedding. *Proceedings of the International Symposium on Information Theory, 2004.*, pages 31–37, June 2004.

[169] E. Semsar-Kazerooni and K. Khorasani. Multi-agent team cooperation: A game theory approach. *Automatica*, 45(10):2205–2213, 2009.

[170] Sanjeev Kumar and Philip R. Cohen. Towards a fault-tolerant multi-agent system architecture. *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 459–466, 2000.

[171] Ted Senator. Science of artificial intelligence and learning for open-world novelty (SAIL-ON). *Defense Advanced Research Projects Agency*, 2019.

[172] DJ Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data. *Advances in Neural Information Processing Systems*, 34:14502–14515, 2021.

[173] Annie Chen, Archit Sharma, Sergey Levine, and Chelsea Finn. You only live once: Single-life reinforcement learning. *Advances in Neural Information Processing Systems*, 35:14784–14797, 2022.

[174] Justyna Karakiewicz. Interventions in complex urban systems: How to enable modeling to account for disruptive innovation. *Understanding Complex Urban Systems: Integrating Multidisciplinary Data in Urban Models*, pages 113–127, 2016.

[175] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.

[176] Jean-Michel Lasry and Pierre-Louis Lions. Mean field games. *Japanese Journal of Mathematics*, 2:229–260, 2007.

[177] Thomy Phan, Thomas Gabor, Andreas Sedlmeier, Fabian Ritz, Bernhard Kempter, Cornel Klein, Horst Sauer, Reiner Schmid, Jan Wieghardt, Marc Zeller, and Claudia Linnhoff-Popien. Learning and testing resilience in cooperative multi-agent systems. *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, page 1055–1063, 2020.

[178] Bart De Schutter Olaf Gietelink, Jeroen Ploeg and Michel Verhaegen. Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations. *Vehicle System Dynamics*, 44(7):569–590, 2006.

[179] Alon Farchy, Samuel Barrett, Patrick MacAlpine, and Peter Stone. Humanoid robots learning to walk faster: from the real world to simulation and back. *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*, page 39–46, 2013.

[180] Valentino Crespi, Aram Galstyan, and Kristina Lerman. Top-down vs bottom-up methodologies in multi-agent system design. *Autonomous Robots*, 24(3):303–313, 2008.

[181] Ali Dorri, Salil S. Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *IEEE Access*, 6:28573–28593, 2018.

[182] BRYAN HORLING and VICTOR LESSER. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(4):281–316, 2004.

[183] H. Hanna. Decentralized approach for multi-robot task allocation problem with uncertain task execution. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 535–540, 2005.

[184] Yuxiao Chen, Ugo Rosolia, and Aaron D. Ames. Decentralized task and path planning for multi-robot systems. *IEEE Robotics and Automation Letters*, 6(3):4337–4344, 2021.

[185] Siddharth Mayya, Diego S. D'antonio, David Saldaña, and Vijay Kumar. Resilient task allocation in heterogeneous multi-robot systems. *IEEE Robotics and Automation Letters*, 6(2):1327–1334, 2021.

[186] Gennaro Notomista, Siddharth Mayya, Yousef Emam, Christopher Kroninger, Addison Bohannon, Seth Hutchinson, and Magnus Egerstedt. A resilient and energy-aware task allocation framework for heterogeneous multirobot systems. *IEEE Transactions on Robotics*, 38(1):159–179, 2022.

[187] Brian Reily, Terran Mott, and Hao Zhang. Adaptation to team composition changes for heterogeneous multi-robot sensor coverage. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9051–9057, 2021.

[188] Donald J Hejna III, Pieter Abbeel, and Lerrel Pinto. Task-agnostic morphology evolution. *The International Conference on Learning Representations*, 2021.

[189] Gabriel Bravo-Palacios, A. Del Prete, and Patrick M. Wensing. One robot for many tasks: Versatile co-design through stochastic programming. *IEEE Robotics and Automation Letters*, 5(2):1680–1687, 2020.

[190] Snehasis Sahoo and Cheng-Yao Lo. Smart manufacturing powered by recent technological advancements: A review. *Journal of Manufacturing Systems*, 64:236–250, 2022.

[191] Xiaolong Ke, Yongheng Yu, Kangsen Li, Tianyi Wang, Bo Zhong, Zhenzhong Wang, Lingbao Kong, Jiang Guo, Lei Huang, Mourad Idir, Chao Liu, and Chunjin Wang. Review on robot-assisted polishing: Status and future trends. *Robotics and Computer-Integrated Manufacturing*, 80:102482, 2023.

[192] Matevž Resman, Maja Turk, and Niko Herakovič. Methodology for planning smart factory. *Procedia CIRP*, 97:401–406, 2021. 8th CIRP Conference of Assembly Technology and Systems.

[193] Feng Li, T.W. Liao, and Lin Zhang. Two-level multi-task scheduling in a cloud manufacturing environment. *Robotics and Computer-Integrated Manufacturing*, 56:127–139, 2019.

[194] Aydin Nassehi Andrew Ma and Chris Snider. Anarchic manufacturing. *International Journal of Production Research*, 57(8):2514–2530, 2019.

[195] Xingyu Li, Aydin Nassehi, and Bogdan I. Epureanu. Degradation-aware decision making in reconfigurable manufacturing systems. *CIRP Annals*, 68(1):431–434, 2019.

[196] Bogdan I Epureanu, Xingyu Li, Aydin Nassehi, and Yoram Koren. An agile production network enabled by reconfigurable manufacturing systems. *CIRP Annals*, 70(1):403–406, 2021.

[197] Yun Geon Kim, Seokgi Lee, Jiyeon Son, Heechul Bae, and Byung Do Chung. Multi-agent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system. *Journal of Manufacturing Systems*, 57:440–450, 2020.

[198] T. W. Liao Feng Li, Lin Zhang and Yongkui Liu. Multi-objective optimisation of multi-task scheduling in cloud manufacturing. *International Journal of Production Research*, 57(12):3847–3863, 2019.

[199] Marc-André Dittrich and Silas Fohlmeister. Cooperative multi-agent system for production control using reinforcement learning. *CIRP Annals*, 69(1):389–392, 2020.

[200] M. Braquet and E. Bakolas. Greedy decentralized auction-based task allocation for multi-agent systems. *Modeling, Estimation and Control Conference (MECC)*, 2021.

[201] Xi Gu and Yoram Koren. Mass-individualisation – the twenty first century manufacturing paradigm. *International Journal of Production Research*, 60(24):7572–7587, 2022.

[202] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda. Cyber-physical systems in manufacturing. *CIRP Annals*, 65(2):621–641, 2016.

[203] Xingyu Li, Yoram Koren, and Bogdan I Epureanu. Complementary learning-team machines to enlighten and exploit human expertise. *CIRP Annals*, 71(1):417–420, 2022.

[204] Andrew Ma, Marcus Frantzén, Chris Snider, and Aydin Nassehi. Anarchic manufacturing: Distributed control for product transition. *Journal of Manufacturing Systems*, 56:1–10, 2020.

[205] Marcello Colledani and Alessio Angius. Integrated production and reconfiguration planning in modular plug-and-produce production systems. *CIRP Annals*, 68(1):435–438, 2019.

[206] Xingyu Li, Alparslan Emrah Bayrak, Bogdan I. Epureanu, and Yoram Koren. Real-time teaming of multiple reconfigurable manufacturing systems. *CIRP Annals*, 67(1):437–440, 2018.

[207] Xingyu Li, Nan Li, Ilya Kolmanovsky, and Bogdan I. Epureanu. Stochastic model predictive control for remanufacturing system management. *Journal of Manufacturing Systems*, 59:355–366, 2021.

[208] Aydin Nassehi, Ray Y. Zhong, Xingyu Li, and Bogdan I. Epureanu. Chapter 11 - review of machine learning technologies and artificial intelligence in modern manufacturing systems. *Design and Operation of Production Networks for Mass Personalization in the Era of Cloud Technology*, pages 317–348, 2022.

[209] Bogdan I. Epureanu, Xingyu Li, Aydin Nassehi, and Yoram Koren. Self-repair of smart manufacturing systems by deep reinforcement learning. *CIRP Annals*, 69(1):421–424, 2020.

[210] Xiaohan Wang, Lin Zhang, Yongkui Liu, Feng Li, Zhen Chen, Chun Zhao, and Tian Bai. Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning. *Journal of Manufacturing Systems*, 65:130–145, 2022.

[211] Xi Vincent Wang, Zsolt Kemény, József Váncza, and Lihui Wang. Human–robot collaborative assembly in cyber-physical production: Classification framework and implementation. *CIRP Annals*, 66(1):5–8, 2017.

[212] Maria Chiara Magnanini, Matteo Mastrangelo, and Tullio A.M. Tolio. Hybrid digital modelling of large manufacturing systems to support continuous evolution. *CIRP Annals*, 71(1):389–392, 2022.

[213] Júlia Bergmann, Dávid Gyulai, and József Váncza. Adaptive agv fleet management in a dynamically changing production environment. *Procedia Manufacturing*, 54:148–153, 2021. 10th CIRP Sponsored Conference on Digital Enterprise Technologies (DET 2020) – Digital Technologies as Enablers of Industrial Competitiveness and Sustainability.

[214] Andrew L Beam and Isaac S Kohane. Big data and machine learning in health care. *Jama*, 319(13):1317–1318, 2018.

[215] Ch Anwar ul Hassan, Jawaid Iqbal, Saddam Hussain, Hussain AlSalman, Mogeeb AA Mosleh, and Syed Sajid Ullah. A computational intelligence approach for predicting medical insurance cost. *Mathematical Problems in Engineering*, 2021:1–13, 2021.

[216] Hua Li, Yumeng Cao, Siwen Li, Jianbin Zhao, and Yutong Sun. Xgboost model and its application to personal credit evaluation. *IEEE Intelligent Systems*, 35(3):52–61, 2020.

[217] Suresh Venkatasubramanian and Mark Alfano. The philosophical basis of algorithmic recourse. *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 284–293, 2020.

[218] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. Face: Feasible and actionable counterfactual explanations. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, page 344–350, 2020.

[219] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, Yuichi Ike, Kento Uemura, and Hiroki Arimura. Ordered counterfactual explanation by mixed-integer linear optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11564–11574, 2021.

[220] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

[221] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 108:895–905, 26–28 Aug 2020.

[222] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 650–665, 2021.

[223] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, page 607–617, 2020.

[224] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. Dace: Distribution aware counterfactual explanation by mixed-integer linear optimization. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 2855–2862, 7 2020.

[225] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. *Parallel Problem Solving from Nature – PPSN XVI*, pages 448–469, 2020.

[226] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Yuichi Ike. Counterfactual explanation trees: Transparent and consistent actionable recourse with decision trees. *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, 151:1846–1870, 2022.

[227] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv*, 2019.

[228] Alejandro Barredo-Arrieta and Javier Del Ser. Plausible counterfactuals: Auditing deep learning classifiers with realistic adversarial examples. *International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020.

[229] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, page 10–19, 2019.

[230] Shohei Shimizu, Takanori Inazumi, Yasuhiro Sogawa, Aapo Hyvärinen, Yoshinobu Kawahara, Takashi Washio, Patrik O. Hoyer, and Kenneth Bollen. Directlingam: A direct method for learning a linear non-gaussian structural equation model. *Journal of Machine Learning Research*, 12(33):1225–1248, 2011.

[231] Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv:1912.03277*, 2019.

[232] Philip Naumann and Eirini Ntoutsi. Consequence-aware sequential counterfactual generation. *Machine Learning and Knowledge Discovery in Databases. Research Track*, 12976:682–698, 2021.

[233] Sohini Upadhyay, Shalmali Joshi, and Himabindu Lakkaraju. Towards robust and reliable algorithmic recourse. *Advances in Neural Information Processing Systems*, 2021.

[234] Duy Nguyen, Ngoc Bui, and Viet Anh Nguyen. Distributionally robust recourse action. *The Eleventh International Conference on Learning Representations*, 2023.

[235] Ronal Singh, Tim Miller, Henrietta Lyons, Liz Sonenberg, Eduardo Velloso, Frank Vetere, Piers Howe, and Paul Dourish. Directive explanations for actionable explainability in machine learning applications. *ACM Transactions on Interactive Intelligent Systems*, 34, 2023.

[236] Ziheng Chen, Fabrizio Silvestri, Jia Wang, He Zhu, Hongshik Ahn, and Gabriele Tolomei. Relax: Reinforcement learning agent explainer for arbitrary predictive models. *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22)*, page 252–261, 2022.

[237] Matthew J Sobel. Mean-variance tradeoffs in an undiscounted mdp. *Operations Research*, 42(1):175–183, 1994.

[238] LA Prashanth and Mohammad Ghavamzadeh. Variance-constrained actor-critic algorithms for discounted and average reward mdps. *Machine Learning*, 105:367–417, 2016.

[239] Yingjie Fei, Zhuoran Yang, and Zhaoran Wang. Risk-sensitive reinforcement learning with function approximation: A debiasing approach. *Proceedings of the 38th International Conference on Machine Learning*, 139:3198–3207, 18–24 Jul 2021.

[240] Vivek S Borkar. Learning algorithms for risk-sensitive control. *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems–MTNS*, 5(9), 2010.

[241] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. *Advances in neural information processing systems*, 28, 2015.

[242] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. A survey of algorithmic recourse: Contrastive explanations and consequential recommendations. *ACM Computing Surveys*, 55(5):1–29, 2022.

[243] Tsvetanka Karagyozova. Module 1: What is risk? 4.4 lower partial standard deviation, 2016. Accessed: 2023-12-10.

[244] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[245] Hans Hofmann. Statlog (German Credit Data). UCI Machine Learning Repository, 1994. DOI: https://doi.org/10.24432/C5NC77.

[246] Kaggle. German credit risk - uci machine learning. https://www.kaggle.com/datasets/uciml/german-credit, 2016. Accessed: 2023-06-18.

[247] Shubham Sharma, Jette Henderson, and Joydeep Ghosh. Certifai: A common framework to provide explanations and analyse the fairness and robustness of black-box models. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, page 166–172, 2020.

[248] Aparajita Haldar, Teddy Cunningham, and Hakan Ferhatosmanoglu. Raguel: Recourse-aware group unfairness elimination. *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, page 666–675, 2022.

[249] J. von Kügelgen, A.-H. Karimi, U. Bhatt, I. Valera, A. Weller, and B. Schölkopf. On the fairness of causal algorithmic recourse. *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, 9:9584–9594, 2022.

[250] Francesca E. D. Raimondi, Andrew R. Lawrence, and Hana Chockler. Equality of effort via algorithmic recourse, 2022.

[251] Haojun Zhu. Predicting earning potential using the adult dataset, 2016.

[252] Patrick E McKnight and Julius Najab. Mann-whitney u test. *The Corsini encyclopedia of psychology*, pages 1–1, 2010.

[253] Anirban Datta. Us health insurance dataset - kaggle. https://www.kaggle.com/datasets/teertha/ushealthinsurancedataset, 2016. Accessed: 2023-08-14.