Coordinating Heterogeneous Teams for Urban Search and Rescue

Zongyue Zhao

CMU-RI-TR-22-44

August 11, 2022

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Dr. Katia Sycara, *chair*
Dr. Changliu Liu
Tejus Gupta

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my family.

Abstract

The mission of Urban Search and Rescue (USAR) is a fight against time. Victim survival depends on whether they can be found and attended to in a critical time frame. However, rescuers face severe visibility issues in the complex environment shortly after structural collapse catastrophes. This brings the need to study effective schemes of team coordination. In this thesis, we present methods to coordinate a rescue team of members with different domain knowledge and capabilities. We first formalize a framework for modeling agents-environment interaction that allows for arbitrary origins of heterogeneity, from which we identify two configuration instances generalizable to common real-world tasks. We use them to develop a series of USAR environment simulators with regard to the trade-off between fidelity and sample efficiency. Under these preparations, we propose a multi-agent reinforcement learning algorithm to tackle USAR. We adopt graph attention, in a novel manner, to fuse information perceived across agents and exploit structural priors of the environment. We apply action-dominant agent indexing to benefit from the power of parameter sharing, while still allowing agents to have different behavioral traits. We show that our proposed approach outperforms previous state-of-the-art literature by 40% to 120% in terms of victim evacuation. In addition, we develop hierarchical planning-based agents that mimic human behavior. We conduct imitation learning over faux human trajectories and demonstrate the improvement over purely online reinforcement learning. Ultimately, we show that the feature distributions of artificial and real data are sufficiently close, so that the former can be used to forecast human performance. We observe that the inference error can be reduced by half when transformer-based predictors are augmented with a synthetic dataset.

Acknowledgments

I would like to express gratitude to my research advisor, Professor Katia Sycara, for her precious insights, guidance, and support. I had an amazing time over the last two years in the lab, in which she created a caring and collaborative atmosphere. I am also thankful to my thesis committee members, Professor Changliu Liu and Tejus Gupta, who have patiently devoted their time to providing valuable feedback and comments.

I also want to say thank you to my collaborators: Professor Michael Lewis, Dr. Dana Hughes, Dr. Joseph Campbell, Dr. Simon Stepputtis, Sophie Yue Guo, Max Chris, Ini Oguntola, Huao Li, Keyang Zheng, Noel Chen, Akshay Dharmavaram, Ying Chen, Ruiyu Li, and Renos Zabounidis, for all the sparkling ideas and comments they kindly shared in our meetings. Special thanks to Joe and Simon for how they put an extensive amount of effort into helping me refine my thesis.

It has been a fantastic journey at CMU. I appreciate the friends I made along the way.

Funding

This work was supported by DARPA Award HR001120C0036. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Earthquakes claimed more than 700,000 lives in the first two decades of this century [28]. Urban areas, in particular, are severely impacted due to extended building structures [97]. Victims under the collapsed ruins must be attended to in a critical time frame for successful evacuation [91]. However, the rescue team, often limited in human resources, faces an unknown, complex environment with hazards associated with secondary disasters [54, 59]. Thus, it is crucial to search for effective schemes of team coordination in urban search and rescue (USAR) scenarios.

We first consider the goal of finding the optimal team policy, *i.e.*, what the rescuers *should* do. For this purpose, we adopt the paradigm of cooperative multi-agent reinforcement learning (MARL), which aims to maximize cumulative reward signals from the interaction between a group of agents and the environment. However, two challenges arise when applying MARL to USAR. Firstly, a typical USAR mission involves intrinsically different operations, such as search, locate, on-site medical support, and extrication [97]. Entities specialized in one domain may not have the knowledge or ability to execute other operations. Thus, the team is *heterogeneous*. Meanwhile, USAR rescuers often encounter severe visibility limitations [87, 121, 126], highlighting the importance of information sharing among teammates [16, 22, 44, 64, 93]. This raises the question of role-adaptive communication: agents need to identify and communicate transferable knowledge while preserving their own behavioral traits.

Recent advances in cooperative MARL realize information sharing by averaging recurrent communication channels [93, 98] or applying the attention mechanism over

the agent relationship graph [10, 55, 58, 70, 90]. However, we observed that the former approach had trouble distinguishing transferable knowledge from domain knowledge only applicable to the sender. The latter approach, on the other hand, is only applicable for tasks involving a great number of agents. Identifying which agent pairs carry more weight in communication becomes of less significance for a small rescue team with only a few agents.

In addition to the methodology of explicit communication, recent literature also adopted the parameter sharing (PS) paradigm that reuses policy or value networks for multiple agents. While PS have been increasingly popular, they are often proposed for homogeneous teams [9, 34, 114]. Work that aims to address this limitation generally relies on agent indexing [26, 63, 83]. However, plain agent indexing fails to utilize the prior knowledge of an agent's capabilities, which are available in various high-level decision-making tasks. Another approach to address issues associated with partial observability is to assume that global states are available for the agents during training [63, 83, 94, 123]. This is known as a variant of centralized training decentralized execution (CTDE). While such global information can be exploited to stabilize learning, acquiring true states is often only feasible in simulated games like StarCraft II [86]. For USAR missions, even if we make simulators where global states are available, agents trained with the state-based CTDE paradigm cannot be further fine-tuned in real-world testbeds, eventually harming the performance in execution [21, 125].

In this thesis, we propose a novel MARL architecture to address the aforementioned drawbacks. Unlike previous literature that operates on the agent relationship graph, we apply attention over the environment graph structure. From a USAR perspective, this corresponds to the case where participants are equipped with a map describing the building structure before the catastrophe, and hope to use outdated priors to expedite the search process. We use a graph attention network [106] to fuse observations shared across agents, without the need to depend on global states. We also improve agent indexing with action availability indicators, so that actor inputs have a higher similarity when the optimal joint action is more homogeneous. Furthermore, we develop a graph-based USAR environment simulator optimized for sample and learning efficiency. We use it to evaluate the performance of the proposed algorithm against state-of-the-art literature in MARL, and show that we outperform methods that use

agent-relationship graphs or demand global information.

While it is beneficial to exploit heterogeneity explicitly presented to the policy network, there exist another series of decision-making tasks in which such extra information is unavailable. For low-level control tasks, it is typical for the action space to appear identical for all agents despite their intrinsic differences. These implicit diversities can be modeled with role-specific environment dynamics. To study the practices applicable to this setup, we develop a grid-based environment also for USAR scenarios. This environment comes with a resolution that maps to 1m $\times$ 1m squares in the real world, effectively allowing for in-room navigation.

As the nature of the grid-based environment demands agents to conquer ambiguity in the action space, we observed that online MARL failed to properly comprehend true short-term objectives that differ for each role. For such a complex and unstructured task, imitation learning (IL) and offline RL serve as two natural replacements for online RL [73, 78, 115]. Compared to IL, offline RL demands a diverse, uniformly distributed dataset that covers the entire state-action space, and poses a higher requirement for the amount of samples [25, 31]. This makes offline RL a less preferable solution in our grid-based environment with high fidelity but low sample efficiency. Meanwhile, (purely offline) IL requires high-performing expert demonstrations to learn from [41, 85]. Because acquiring human data for USAR tasks is costly and time-intensive [19, 29, 32, 77], we develop planning-based faux human agents, a practice adopted by modern literature in complex tasks like autonomous driving [56, 111, 122]. We demonstrate the empirical advantages of imitating experts over online RL.

In addition to the study on optimal coordination, we also hope to comprehend human behavior in USAR, *i.e.*, what the rescuers *currently* do. Understanding so allows assistive robots to cope with human behavior or issue adequate intervention at an early stage, eventually improving the mission outcome. In the field of human-robot collaboration, it is common to build cognitive models that predict human intent [12, 81, 120] or performance [13, 18, 36]. Deep neural networks are progressively assuming a crucial role for these purposes [89]. For example, recent literature has adopted sequential models like RNNs [61, 112] and transformers [43, 52, 69]. These architectures are known to demand a vast amount of training data [80, 110], sometimes more than 10^{10} annotated data points [46]. As it is almost impossible to collect such an amount of human data for USAR missions, we consider the application of using

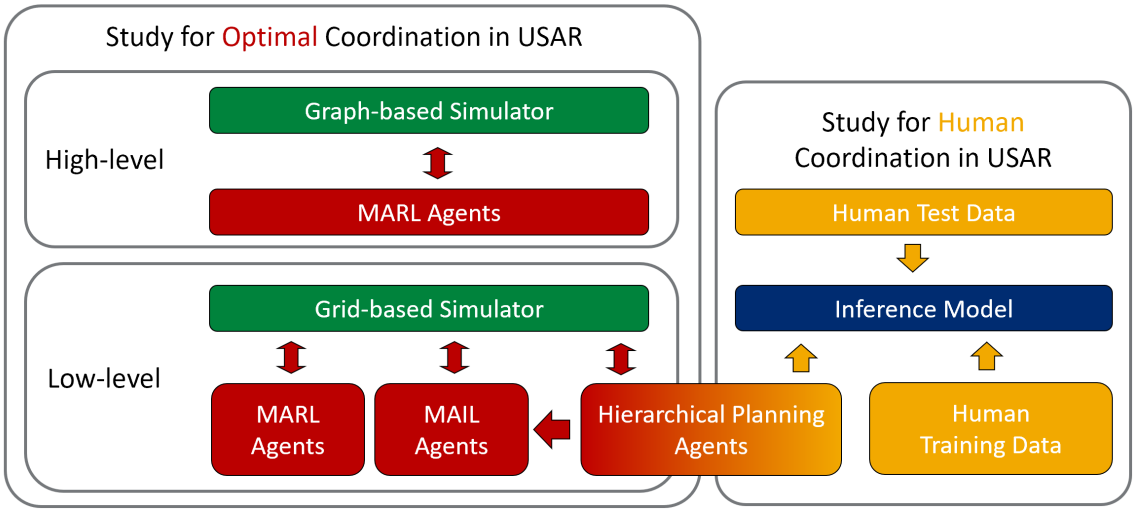


Figure 1.1: Study Outline

artificial trajectories for inference tasks in USAR. We tune our planning-based agents to show behavioral traits similar to real humans, and use faux-human trajectories to help train a transformer-based prediction model. We demonstrate advantages, in terms of inference accuracy over unseen human data, against training without synthetic data. We also show how such a data-intensive architecture is proper for the task by showing improvements against simpler models.

To conclude, the outline of our study is summarized in Figure 1.1. The rest of this thesis is organized correspondingly as follows:

- In Chapter 2, we review the key concepts in multi-agent reinforcement learning and provide preliminaries for our work.
- In Chapter 3, we formalize a universal framework to model heterogeneity in cooperative MARL and identify two representative configurations. We also provide an overview of the USAR mission we aim to address.
- In Chapter 4, we present a graph-based environment to simulate USAR at a high level. We propose an online MARL architecture and show improvement against previous state-of-the-art algorithms designed for generic setups.
- In Chapter 5, we present a grid-based environment to simulate USAR at a low level. We show the limitation of online MARL towards action ambiguity. We demonstrate how imitating vivid faux humans may improve decision-making

given the same network architecture. We also show that synthetic trajectories can be used to augment cognitive models that aim to infer human performance.

- In Chapter 6, we conclude the findings in this thesis.
- In Appendix A, we describe our collaborators' methodology when acquiring human data.
- In Appendix B, we present implementation details and acknowledge aspects for future work.

Chapter 2

Background

2.1 Markov Decision Process

In fully-observable, single-agent reinforcement learning, the interaction between the agent and the environment is formalized as a Markov decision process (MDP) [101], which is represented by the Markov tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$:

- The state space $\mathcal{S}$ is the set of possible states of the environment.
- The action space $\mathcal{A}$ consists of the possible actions the agent may take.
- The transition function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ determines the probability $T(s, a, s')$ of the environment arriving in the state s' after the agent executed the action a given state s. The one-step Markov property is attained, as the entire dynamics of the environment are described with the triplet (s, a, s') but not history.
- The reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ maps the state s, action a, and the resulting state s' to an reward signal $R(s, a, s')$ known by the agent after reaching s'.
- The discount factor $\gamma \in [0, 1]$ weighs the importance of future rewards over immediate rewards.

Under MDP, a policy $\pi(a|s)$ is considered optimal if it maximizes the expected discounted cumulative rewards $G = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t]$ [79]. If the transition dynamics T

and reward function R are fully known, the optimal policy can be solved via dynamic programming [6]. Otherwise, RL algorithms need to learn from agent-environment interaction experiences in an error-and-trial manner. Furthermore, MDP assumes complete access to the environment states. If the inception capabilities of the agent are limited, decision-making can be modeled as a partially observable Markov decision process (POMDP) by introducing the observation space $\Omega = \{o\}$ and the observation function $\mathcal{O} : \mathcal{S} \times \mathcal{A} \times \Omega \to [0, 1]$ [96]. As the probability $p(o'|o, a)$ does not represent the environment dynamics, algorithms designed for POMDP commonly hold a belief regarding the unseen environment states [68, 96] or use dynamic models [39, 95].

2.2 Markov Games

In this section, we discuss how to model the interaction between multiple agents and the environment. Among the various formulations proposed to extend (PO)MDPs to a multi-agent setting, the two key differences are a) which elements in the Markov tuple are agent-sensitive, and b) the paradigms that regularize and reflect the information structure among agents.

2.2.1 Heterogeneity in Multi-Agent Systems

Consider the partially observable Markov tuple. A generalizable adaptation for the multi-agent setting is to replace selected elements $(\mathcal{X}, \mathcal{Y}, ...) \subseteq (\mathcal{S}, \mathcal{A}, \Omega, T, R, \mathcal{O}, \gamma)$ with sets of those elements $(\{\mathcal{X}^{(i)}\}, \{\mathcal{Y}^{(i)}\}, ...)$ to address heterogeneity between agents. For example, the Markov Games model [57, 92] allows for a set of action spaces $\{\mathcal{A}^{(i)}\}$ and reward functions $\{R^{(i)}\}$. Under this formulation, it is possible to define whether a task is **cooperative** based on the reward structure: either a) the same reward function applies to all agents ($\forall i, j, R^{(i)} = R^{(j)}$, also known as the MMDP model [8]), or b) the overall optimization target is the average of agent-specific returns ($\max \sum_i G^{(i)}$) [47, 118]. Furthermore, there exists work that considered heterogeneity in Markov elements other than $\mathcal{A}$ and R. The POSG model [35] supports disjoint observation spaces $\{\Omega^{(i)}\}, \{\mathcal{O}^{(i)}\}$ in addition to action space and rewards variations. As discussed by the authors of [35], when the reward function is uniform across agents, POSG regresses to DEC-POMDP [7], which is a widely adopted framework in recent

work on deep cooperative MARL [11, 23, 27, 83]. Moreover, the MAH-POMDP [90] model extended DEC-POMDP with state spaces $\{\mathcal{S}^{(i)}\}$ that are unique to the agent's corresponding class.

2.2.2 Information Structure

The second thing to consider when formalizing cooperative MARL tasks is how agents are allowed, in both explicit regulations and feasibility constraints, to share information among themselves [119]. Certain real-world scenarios, *e.g.*, when agents are geometrically sparse [76] or facing adversarial attacks [99], constrain or even prohibit timely information exchange. Under the most strict limitation, MARL can be realized in a fully decentralized setting [7, 118], where each agent corresponds to an independent policy that only considers its own observation. Algorithms in this setup scale linearly with the number of agents. However, for any arbitrary agent i, because the environment evolves due to not only its own action but also other agents' actions unknown to i, the transition dynamics perceived by i is non-stationary [71]. This violation to the Markov property means agent i's value estimate may not approximate the status quo of the environment as other agents' learning progresses.

Meanwhile, suppose the task scenario can be separated into distinct training and execution stages, where information exchange is only prohibited during execution. In that case, it is possible to mitigate the aforementioned deficiencies via the paradigm of centralized training with decentralized execution (CTDE). This paradigm has been popular in recent advances [14, 53, 60, 63, 83, 84, 94] in cooperative MARL. The problem setup is similar to DEC-POMDP, where agents make decisions under their own observation/action space rather than joint spaces. However, during the training stages, participants are allowed to freely share information with each other [34, 60] or access global states [83, 86]. Such extra information is often only used to train the critic in the actor-critic setting (Section 2.3.2), so that the actor heads face the same input distribution in both stages. Nonetheless, the environment under this paradigm would appear more stationary to the agents, leading to performance improvements against non-CTDE baselines in certain tasks [15] that allow for a training playground.

Parameter sharing (PS) [103] is the technique where multiple agents reuse a policy and/or value network. This mechanism abides by the same constraints regarding

information flow as CTDE. So long as there exists a centralized playground at the training stage, trajectories from all agents can be gathered to update the agent models, whether they share weights (PS) or not (Non-PS CTDE). The trained models can be deployed separately to each agent prior to the execution stage, thus fulfilling the requirement of decentralized execution. For cooperative settings, it has been observed that this mechanism facilitates training speed by reducing the total amount of trainable parameters and improves converged returns by utilizing instantaneous peer information and learned knowledge [15, 48]. Unfortunately, a fully shared model encounters issues adapting to multiple coherently distinct tasks when used naively. Thus, it is common to condition the network with a one-hot vector indicating the agent's identity [26, 34, 63, 83, 104]. This method is known as agent indexing or agent identification. An alternative solution is to share partially, *e.g.*, sharing the critic but keeping actor heads separated in an actor-critic framework [17, 117]. For tasks that involve a large number of agents, it is also possible to use more than one shared network. Agents who most closely resemble each other can be grouped together and assigned with the same model [15, 107].

In scenarios that pose no constraints on the information flow, MARL can be realized by adopting a single policy that maps the joint observation $o \in \prod_i^n \mathcal{O}^{(i)}$ of all agents to a joint action $a \in \prod_i^n \mathcal{A}^{(i)}$ [2]. This method allows for reusing existing single-agent RL algorithms, and is immune to non-stationarity resulted from independent learners with partial observability. However, it suffers from scalability issues, as the observation and action space of the joint controller scale exponentially with respect to the number of agents [34]. This calls for a representation of semi-independent learners that corresponds to agent-specific spaces. In addition to the CTDE/PS paradigms that are still applicable, agents here may adopt explicit communication mechanisms via recurrent channels [58, 74], attention [42, 45, 90], or shared memories [75]. There is also a series of works that aim to factorize value networks of the centralized controller, which is discussed in detail in Section 2.3.1.

2.3 Reinforcement Learning Algorithms

In this section, we briefly review key reinforcement learning algorithms and the special considerations when multiple agents are involved.

2.3.1 Value-based

Action-value methods maintain value estimates for state-action pairs and use them to select actions. For example, Q-Learning [108] estimate the optimal state-action value function Q from rollouts $(s, a \sim \pi(\cdot|s), r, s')$ by any policy π, commonly a ε-soft variant of the greedy policy represented by Q:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{2.1}$$

For tasks with high-dimensional or continuous state/action spaces, neural networks can be used as function approximators for the Q-value. With the trainable parameters in the network denoted as θ, the loss function to be minimized can be expressed as:

$$L(\theta) = \mathbb{E}_{\pi} \left[\left(r + \gamma \max_{a'} Q(s',a'|\theta) - Q(s,a|\theta) \right)^2 \right] \tag{2.2}$$

To address instability issues from a long and correlated decision sequence, a popular architecture, DQN [66] introduced the replay buffer $D_t = \{e_1, ..., e_t\}$, where $e_t = (s_t, a_t, r_t, s_{t+1})$ is the Markov tuple of the past experience at t. During training, gradient descent is conducted on minibatches sampled from the uniform distribution of the replay buffer: $\{(s,a,r,s')\} \sim U(D_t)$, instead of the current Markov tuple (s_t, a_t, r_t, s_{t+1}) only. It was observed that replay buffers, together with periodical updates of the target Q function, helped DQN to achieve human-equivalent performance in Atari [5].

DQN and other off-policy algorithms with a replay buffer encounter issues when extended to the multi-agent setup. As discussed in Section 2.2.2, the joint state and action space would scale exponentially when trained with a centralized controller, challenging the expression power of the Q-network. Meanwhile, Q-networks in the fully decentralized approach (independent Q-learning [103]) would face a non-stationary environment, so optimal convergence cannot be guaranteed. The non-stationarity introduced by independent agents also invalidates the use of a replay buffer [60]. As discussed in Section 2.2.2, the transition dynamics $T(s'|s,a)$ for an agent i depends on the policy of all other agents $\forall j \neq i$. As the policies π_j progress independently over time, even for a replay buffer D^i that only contains experience from the current

agent, the dynamics reflected by the sampled batch $\{(s, a, r, s')\}$ no longer match the current environment.

For cooperative tasks that issue a uniform reward signal for the entire team, even when the number of agents is small and centralized learning can be adopted to mitigate the non-stationarity issue, value-based algorithms still encounter the "lazy-learner" problem. After the centralized value network is learned such that it performs well for one agent, the exploration of other agents will be suppressed because doing so would worsen the team reward [100]. Thus, VDN [100] has been proposed to decompose the centralized Q-value estimation Q_{tot} into a sum of individual value functions $\{Q_a\} : \sum_a Q_a = Q_{tot}$, each corresponding to an individual agent. QMix [83] improved the art of factorization by introducing a mixing network instead of simple summation. As long as monotonicity is preserved between Q_{tot} and $\{Q_a\}$, greedy actions based on each individual value network would still be equivalent to the greedy joint action based on Q_{tot}. Meanwhile, the mixing network allows for learnable non-linear factorization, resulting in a significant performance boost in the StarCraft II Environment [86].

2.3.2 Policy Gradient and Actor-Critic

Policy gradient algorithms directly optimize the parametrized policy $\pi(a|s, \theta)$ to maximize the discounted return $J(\theta) = \mathbb{E}_{\pi(\theta)}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$, by stepping θ into the direction of the gradient $\nabla_\theta J(\theta)$, which can be approximated [102] as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi(\theta)}\left[(\nabla_\theta \log \pi(a|s, \theta))\, Q^\pi(s, a)\right] \tag{2.3}$$

where $\rho^\pi(s) = \lim_{t\to\infty} Pr(s_t = s|s_0, \pi)$ (provided that it exists) is the stationary distribution of states given policy π. Under a differentiable representation of the policy network $\pi(a|s, \theta)$, the problem becomes estimating the action-value $Q^\pi(s, a)$. For example, REINFORCE [109] uses the Monte-Carlo estimate $G_t = \sum_{t=t_0}^{T} \gamma^{t-t_0} r_t$, which is a unbiased estimation of the true action value. Unfortunately, the Monte-Carlo estimation suffers from high variance [51]. The authors of [109] suggested reducing the variance by subtracting G_t with a learned function $b(s_t|\omega)$ known as the

baseline. The estimate of the policy gradient at time s now becomes:

$$\nabla_\theta \log \pi(a_t|s_t, \theta_t) \left(G_t - b(s_t|\omega_t)\right) \tag{2.4}$$

A natural placement for the baseline function is the state-value estimate $V^\pi(s)$. In this case, the scaling factor $G_t - V(s_t|\omega_t)$ can be seen as an estimate of the advantage of action a_t at state s_t, because G_t estimates the action-value $Q(s_t, a_t)$ [101]. In general, mechanisms that involve approximations to both the policy and value functions belong to the family of *actor-critic*, where the actor refers to the policy model and the critic refers to the value estimation. In a stricter definition, actor-critic frameworks use TD-learning to estimate a state-action value model $Q(s, a|\omega)$ in place of the Monte-Carlo estimate G_t, thereby enabling step-wise updates and facilities training [117]. The use of a baseline function $V^\pi(s)$ still applies, leading to the advantage actor-critic (A2C) method [67]. Furthermore, bootstrapping can be applied to the estimation of $V^\pi(s)$, leading to a generalized advantage estimator (GAE) [88].

Actor-critic algorithms have been widely applied in multi-agent tasks. They have a particularly strong presence in tasks that limit information flow during execution and benefit from implicit coordination paradigms (Section 2.2.2). Since the critic is only used during training, they can be designed to take extra information from peer agents as input (MADDPG [60], COMA [27]), or be shared across multiple agents [62, 117]. Meanwhile, tasks that allow explicit communication during the execution stage also commonly adopt policy gradient algorithms. In this aspect, IC3Net [93] uses LSTM units as communication channels that persist across timestamps. G2ANet [58], HetNet [90], MAGAT [55], GAMA [10], and MAGIC [70] form relationship graphs over agents to apply attention-based communication.

2.4 RL over Graph Environments

Our main contributions in Chapter 4 involve applying MARL over a graph-based environment simulator. Thus, we hope to review literature in this domain and discuss the similarities and differences.

The work that most closely resembled ours is [127], which applied a graph attention network (GAT) to solve graph navigation. While we also adopted GAT to examine

neighborhood features, the main differences are as follows:

- The scope of [127] is single-agent RL that aim to one-shot navigate unseen graph structures. The agent's sole purpose is to cover more graph nodes within the given time budget. On the contrary, we focus on coordinating a team of multiple agents. Rapid graph navigation would not necessarily lead to desirable teamwork.
- In [127], a graph node contains no feature other than its identity. Meanwhile, nodes in our graph differ from each other by their content (victims, rubbles, *etc.*). Our graph is thus heterogeneous [116], such that it can be used to study USAR missions that involve not only graph navigation but also how agents interact with the node they reside in.
- We adopted different network architectures and gradient estimation methods.

Recent advances applied RL in other graph-related problems, such as minimum vertex cover, traveling salesman, and maximum cut [4, 40, 49]. These tasks reduce to combinatorial optimization [65], and like the problem investigated in [127], they focus on how identical nodes are connected to each other. On the contrary, our problem statement primarily focus on how graph nodes host information about different types of entities. We argue that this distinction separates our work in Chapter 4, *i.e.*, coordinating heterogeneous agents in a heterogeneous graph, from previous literature.

Chapter 3

Task Formulation

3.1 Heterogeneous Teamwork

The problem of coordinating a heterogeneous team including N agents with $C \leq N$ distinct roles can be formalized by the tuple:

$$(\mathcal{C}, \mathcal{S}, \{\mathcal{A}^{(c)}\}, \{\mathcal{M}^{(c)}\}, \{\Omega^{(c)}\}, \{\mathcal{O}^{(c)}\}, \{\mathcal{T}^{(c)}\}, \{\mathcal{R}^{(c)}\}, \gamma) \tag{3.1}$$

- $\mathcal{C} = \{1, ..., N\} \rightarrow \{1, ..., C\}$, where $C(i)$ indicates the role of agent i. Given the current state, agents with the same role are interchangeable with respect to all other Markov elements.
- $\mathcal{S}$, which is the space of global states. The entire environment is uniquely determined given $\mathcal{C}$ and $\mathcal{S}$, *i.e.*, we do not consider information describing an agent (other than its role) to be separated from $\mathcal{S}$. For example, $s \in \mathcal{S}$ contain information on the location of any arbitrary agent i.
- $\mathcal{A} = \cup\mathcal{A}^{(c)} = \{a\}$ is the union of all agent-wise actions, where $\mathcal{A}^{C(i)}$ is the action space for the i-th agent. This is different from the joint action space $\mathcal{U} = \prod \mathcal{A}^{C(i)} = \{A\}$.
- $\mathcal{M}^{(c)} : \mathcal{S} \times \mathcal{A} \rightarrow \{0, 1\}$, where the action mask $M^{(c)}(s, a) = 1$ indicates that the action a is available for an agent with role c given the global state s. Note how the separation of $\mathcal{C}$ from $\mathcal{S}$ makes $\mathcal{M}$ conditional to c as well.

- $\Omega = \cup\Omega^{(c)} = \{o\}$, which is the union of all role-specific observation spaces. $\Omega \subset \mathcal{S} \cup \mathcal{C}$ is a proper subset of the union of the state space and the set of role mappings.
- $\mathcal{O}^{(c)} : \mathcal{S} \times \Omega \to [0, 1]$ is the observation function, where $O^{(c)}(s, o)$ is the probability of an agent with role c observes o given the global state c. Compared to the DEC-POMDP setting that requires a state-action pair(s, a), this marginalized formalization indicates that agents may observe the environment without the need to execute an action. This is common for modern environments [86] with a high observation rate asynchronous to action selection.
- $\mathcal{T}^{(c)} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition dynamics, where $T^{(c)}(s, a, s')$ is the probability of the environment arriving at s' after an agent with type c executed action a given the previous global state s. Note that this allows for role-specific dynamics on top of the possible constraints on the action space. We find this useful for real-world scenarios where the action space can be decomposed to task-irrelevant atomic actions. Moreover, this assumes sequential action execution, *i.e.*, the environment immediately transits after a single agent calls for action. This allows for a simplified representation of role-specific environment dynamics because it no longer depends on the agents' joint action.
- $\mathcal{R}^{(c)} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, where $R^{(c)}(s, a, s')$ is the reward received by an agent of type c executed the action a from state s resulting in state s'.
- $\gamma \in [0, 1]$ is the discount factor uniform to all agents. We do not consider role-specific discount factors to simplify advantage estimation under parameter sharing.

In this thesis, we consider cooperative tasks only, where the end goal is to maximize the average expected discounted return from all agents:

$$\max \mathbb{E}_{\pi_1,\ldots,\pi_N} \left[\sum_{i=1}^{N} \sum_{t=0}^{T} \gamma^t R_t \right] \tag{3.2}$$

Within this framework, we focus on two configurations that have a significant presence in real-world tasks. The first one, which we will refer to as *micromanagement*, has a uniform action space $\mathcal{A}^{(c)} = \mathcal{A}, \forall c$. In this case, agents differ by the structure of their corresponding transition dynamics $\mathcal{T}^{(c)}$. We call this micromanagement because

this setting is often found in low-level control problems. Consider a set of 6-DOF open-chain manipulators with different dimensions and end-effectors. The action space for each manipulator is the same Cartesian product of six individual joint spaces ranging from zero to two pi. However, the outcome from the interaction between the manipulators and the environment will differ subject to their intrinsic properties (dimensions, tools, *etc.*). This form of heterogeneity can be modeled with either role-specific transition dynamics or states. Here, we adopt the former method to better indicate that the aforementioned intrinsic is stationary over time and episodes.

The second configuration that we consider often corresponds to high-level decision-making tasks, which we will refer to as *macromanagement*. In this case, heterogeneity primarily comes from well-defined disjoint sub-action spaces. For each role, we can decompose its corresponding action space as follows:

$$\mathcal{A}^{(c)} = \mathcal{A}_e^{(c)} \cup \mathcal{A}_o \tag{3.3}$$

where $\mathcal{A}_e^{(c)} \cap \mathcal{A}_o = \varnothing$ and $\mathcal{A}_e^{(m)} \cap \mathcal{A}_e^{(n)} = \varnothing, \forall m \neq n$. $\mathcal{A}_o$ represents role-invariant actions, *e.g.*, navigation, and depends on the state $\mathbf{s}$ only. On the other hand, the role-specific disjoint subspaces $\mathcal{A}_e^{(c)}$ often correspond to explainable agent capabilities declared by the task and environment. Consider a USAR setting, for example. In a rescue team consisting of people from different backgrounds, it is common that only licensed medical practitioners have sufficient knowledge to diagnose and stabilize a victim's condition properly. This stabilization action belongs to $\mathcal{A}_e$ for the medical role. As demonstrated in Section 4.3, information regarding an agent's action space can be used as a strong indicator of the essence of that agent; and should be taken into account of designing heterogeneous autonomous systems. Furthermore, although this configuration does not rule out heterogeneous reward structures $R^{(c)}$, for the sake of simplicity, we assume all roles perform an action equivalently well, provided that the action is in its current action space.

3.2 USAR Mission Design

We consider a USAR mission involving a team of three participants. The end goal of this mission is to evacuate injured victims out of a partially collapsed building. As

Figure 3.1: Saturn map for the USAR mission.

the locations of victims are unknown to the rescue team in advance, the team must learn to efficiently explore the building in a limited amount of time. However, certain hallways and rooms are blocked by rubbles that have collapsed from the building structure. This issue can be mitigated by the *engineer* in the team. After victims are found, their condition needs to be treated on-site before they can be carried out of the building. This requires the presence of the *medic*, the only member with sufficient medical expertise. It is also worth noting that victims require different treatment according to their injury types, either abrasions (A), bone damage (B), or *critical* condition (C). Victims that suffered the former two types of injury are also referred to as *regular* victims, as their condition can be treated by the medic alone. On the contrary, stabilizing a critical victim requires both the medic and another team member to be present. After the pre-condition of stabilization is met, it is the *transporter*'s job to move fast and carry the victim to the correct evacuation zones. Figure 3.1 shows a grid-world representation of the map, which we will refer to as the *Saturn* map, used for this mission.

From the perspective of a USAR mission, the performance of the team should be evaluated by the weighted sum of the number of victims evacuated to the correct zone. The team is rewarded with 10 points for each regular victim it evacuated. Meanwhile, as evacuating a critical victim demands more close collaboration between

the teammates, they are worth 50 points each. For the reinforcement learning agents, these scores are only reflected in the reward function of the participant who is directly responsible for transportation. Thus, we also issue intermediate rewards to help other participants identify desirable behavior, as shown Table 3.1. To distinguish between the RL-reward function and the actual task performance, we will refer to the team score as the M1 metric.

Event	Reward
The engineer cleaned a rubble block.	1.0
The medic stabilized a regular victim	1.0
A participant picked up a stabilized regular victim	1.0
The medic stabilized a critical victim	1.0
A participant picked up a stabilized critical victim	5.0
A participant evacuated a regular victim.	10.0
A participant evacuated a critical victim.	50.0

Table 3.1: Reward signals.

Furthermore, we investigate several variants of the general mission context discussed above. For example, in Chapter 4 only the transporter is allowed to carry an victim. In Chapter 5, all roles may carry an victim, but the maximum moving speed of a participant varies based on their roles: the transporter moves significantly faster than the other participants, whereas the engineer moves the slowest. These variations will be discussed in more detail in the following chapters.

At the end, we make the assumption that communication infrastructure remains functional following the disaster, such that MARL agents may freely share knowledge as they please. Thus, we compare our proposed architecture with previous literature equipped with both implicit and explicit communication mechanisms (in Section 4.3). Moreover, partial observability heavily applies in our task setting. As discussed in the introduction, we provide global states (during training) only for our benchmarking literature, in the manner they see fit.

Chapter 4

Marcomanagement

4.1 Graph-based Environment Simulator

The graph-based environment[1] aims to simulate the USAR task at the room level. Each room or corridor segment is represented with a node in the graph, whereas (non-directional) graph edges show connectivity.

4.1.1 State Space

The global state $\mathbf{s} \in \mathcal{S}$ and agent-role mapping $\mathcal{C}$ are represented with a dictionary with entry sizes:

- `graph`: $\mathbf{s}_g \in \mathbb{R}^{G \times (f_n + N)}$
- `agent`: $\mathbf{s}_a \in \mathbb{R}^{N \times (f_a)}$

where G is the number of nodes, N is the number of agents, and f_n is the dimension encoding a node. $f_a = f_i + C$ where C is the number of roles and f_i is the number of injury types. The `graph` entry encodes the content of each node, including the number of stabilized and unstabilized victims, evacuation zones of each injury type, the number of rubbles, and the whether a certain player is on this node. The `agent` entry encodes information that suffices to describe each agent, including its role and (if applicable) the victim's injury type it is carrying. One-hot encoding is used in these features when appropriate.

[1] https://gitlab.com/cmu_asist/gym_graph

4.1.2 Observation Space

Upon calling the `step` function of the environment, a dictionary with the following optional keys is returned for each agent, *e.g.*, agent i:

- `obs`: $\mathbf{o}^{(i)} \in \Omega$ is a dictionary in the same fashion as $\mathbf{s}$:
  - `graph`: $\mathbf{o}_g^{(i)} \in \mathbb{R}^{G \times f_g}$
  - `agent`: $\mathbf{o}_a^{(i)} \in \mathbb{R}^{f_a}$

  where $f_g \doteq f_n + 1$ is notion of the encoding dimension of each node. The additional dimension indicates whether agent i is in this room. In addition, the `agent` entry only contains information regarding the corresponding agent and thus is one-dimensional. Furthermore, the visibility mask $O(s, \cdot)$ is constructed in a way to mimic the challenging limitations encountered in real-world USAR tasks. All information in the `graph` entry will be masked to zeros except for the agent's current room, effectively making it one-hot along the rows.
- `agent_id`, which is a one-hot vector $\mathbf{i}^{(i)}$ encoding the agent's identity i. This is not used by our model but provided for benchmarking literature that adopt plain agent indexing.
- `action_mask`, which is the multi-hot vector $M^{C(i)}(s, \cdot)$ indicating the agent's current action space. We will use $\mathbf{m}^{(i)} \in \{0, 1\}^{|\mathcal{A}|}$ to denote this vector.

Both the states and observations are normalized to $[0, 1]$ channel-wise.

4.1.3 Action Space and Dynamics

The size of the action space is $|\mathcal{A}| = |\mathcal{A}_o| + |\mathcal{A}_e|$, where $|\mathcal{A}_o| = G + 1$ is the number of actions available to all roles and $|\mathcal{A}_e| \doteq |\cup_c \mathcal{A}_e^{(c)}|$ is the total number of role-specific actions. Among the G navigational actions, only the nodes adjacent to the current room are available. As a node is not considered as its neighbor, an agent must explicitly call the `still` action to keep waiting in the current location. The role-specific actions, which describe interactions between the agent and room contents, include `clean`, `stabilize`, `pick` and `evacuate`. `clean` is unique to the engineer, `stabilize` is unique to the medic, and the last two actions are unique to the transporter. Note that if rubbles exist in a room, they must be cleaned by the engineer before any other

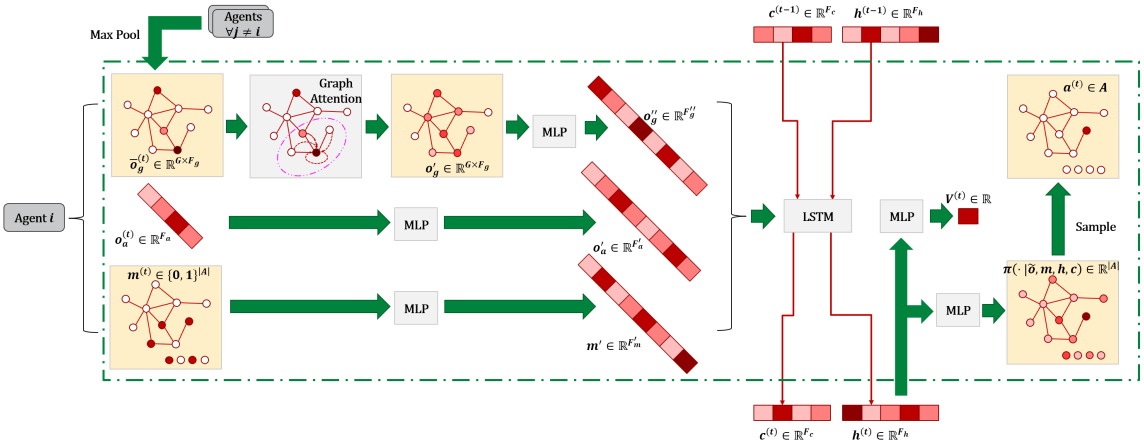


Figure 4.1: Model architecture.

interactions with the room content.

We study the USAR problem where the building structure prior to the disaster is known and fixed, but agents will encounter path-blocking rubbles that have collapsed to random locations. The locations of victims and evacuation zones are also unknown to the agents in advance. These variables, along with the initial locations of agents, will be randomized for each episode to avoid memorization.

4.2 Algorithms

Figure 4.1 shows the proposed actor-critic algorithm. The network is shared among all agents. Superscripts that indicate timestamp or agent identity are omitted where obvious.

At each timestamp t, the environment returns the dictionary:

$$\{i : (\mathbf{o}_g^{(t,i)}, \mathbf{o}_a^{(t,i)}, \mathbf{m}^{(t,i)})\ \forall i\} \tag{4.1}$$

An arbitrary agent i first receives communication $\{\mathbf{o}_g^{(t,j)}\ \forall j \neq i\}$ from all other agents, then merge them with its own observation with max pooling:

$$\bar{\mathbf{o}}_g^{(t,i)} = \max_{1 \leq j \leq N} \left\{\mathbf{o}_g^{(t,j)}\right\} \tag{4.2}$$

The observation scheme described in Section 4.1 ensures that agents will not suffer from outdated knowledge from a peer that failed to comprehend this non-stationary environment, as opposed to memory-based communication like [75]. However, the merged observation is still highly sparse: only N out of G entries in $\bar{\mathbf{o}}_g^{(t,i)}$ are non-trivial. This is an undesirable behavior, as the contents $\bar{\mathbf{o}}_g^{(t,i)}(p) \in \mathbb{R}^{f_g}$ of an arbitrary node p should correlate with the value of not only itself but also its neighborhood $\mathcal{N}(p)$. Therefore, we apply a graph attention network [106] to integrate features over the environment graph.

For node p, the corresponding output feature from the attention layer is:

$$\mathbf{o}_g^{\prime(t,i)}(p) = \sum_{q \in \mathcal{N}(p) \cup \{p\}} \alpha_{p,q} W \bar{\mathbf{o}}_g^{(t,i)}(q) \tag{4.3}$$

where $W \in \mathbb{R}^{f_g \times f_g}$ is a weight matrix shared across this graph attention layer, and the normalized attention coefficient $\alpha_{p,q} \in \mathbb{R}$ between node p and q is computed as:

$$\alpha_{p,q} = \frac{\exp(e_{pq})}{\sum_{r \in \mathcal{N}(p) \cup \{p\})} \exp(e_{pr})} \tag{4.4}$$

where the raw attention coefficient $e_{p,q} \in \mathbb{R}$ is computed with linear weights $\mathbf{a} \in \mathbb{R}^{2f_g \times 1}$:

$$e_{p,q} = \text{LeakyReLU}\left(\mathbf{a} \cdot \left[W \bar{\mathbf{o}}_g^{(t,i)}(p) \| W \bar{\mathbf{o}}_g^{(t,i)}(q)\right]\right) \tag{4.5}$$

which indicate the importance of node q's contents to node p. As adopted by [106], attention is conducted over the first-order neighborhood of p including the node itself, which corresponds to graph-navigation action space $A_o(s^{(t)})$. Compared to literature that compute agent-wise importance scores [42, 58], such a room-wise importance fits better to our task involving merely three agents but a large graph structure.

We also improve plain agent-indexing to attentively identify transferable knowledge. Under the scheme of agent indexing, the policy model is guaranteed to perceive unequal input $\{\mathbf{o}, \mathbf{i}\}$ for different agents, even though the raw observation $[\mathbf{o}^{(i)}, ...]$ from the environment could be similar. The benefit of doing so is that it causes agent-specific hidden states to be deduced within a fully shared policy network, thus allowing for heterogeneous behavioral traits. However, sub-optimality exists in this approach. Suppose an optimal joint policy $\Pi^*(A|\prod_i \mathbf{o}^{(i)})$ exists for our setup. The goal of

training a shared and indexed policy model $\pi_\theta(a|\mathbf{o}, \mathbf{i})$ is to satisfy:

$$\prod_i \pi_\theta \left(\cdot|\mathbf{o}^{(i)}, \mathbf{i}^{(i)}\right) = \Pi^*(\cdot|\prod_i \mathbf{o}^{(i)}) \tag{4.6}$$

Consider the case where agents share similar optimal behavior:

$$\exists \mathbf{s} \in \mathcal{S}, \mathbf{o} \in \bigcap_i \Omega^{C(i)}(\mathbf{s}), a \in \bigcap_i \mathcal{A}^{C(i)}(\mathbf{s}) : \begin{cases} \forall i, \mathbf{o}^{(i)} \in \Omega^{C(i)} & = \mathbf{o} \\ \Pi^* \left(\cdot|\prod_i \mathbf{o}\right) & = \prod_i a \end{cases} \tag{4.7}$$

In this case, despite the joint optimal controller's input consisting of identical components $\mathbf{o}$, the input to the parameterized shared model π_θ still differs for each agent.

$$\forall 1 \leq i, j \leq N : i \neq j, \left\{\mathbf{o}^{(i)}, \mathbf{i}^{(i)}\right\} \neq \left\{\mathbf{o}^{(j)}, \mathbf{i}^{(j)}\right\} \tag{4.8}$$

Furthermore, the component that causes such a difference are agent-wise orthogonal: $\mathbf{i}^{(i)} \perp \mathbf{i}^{(j)}$, which requires the shared model to find adequate weights θ that deactivates $\forall \mathbf{i}$ for all observations-action pairs $(\mathbf{o}, a)$ that satisfy Equation 4.7. While it is possible to approximately do so with a sufficiently large neural network, we argue that this learning paradigm can be improved by conditioning the model with action mask $\mathbf{m}^{(i)}$ that denotes $\mathcal{A}^{C(i)}(\mathbf{s})$. This is because the optimal joint policy may only appear homogeneous at the intersection of role-specific action spaces $\cap_i \mathcal{A}^{C(i)}(\mathbf{s})$. For all cases that satisfy Equation 4.7, inputs to the new model have a greater similarity:

$$\begin{aligned} \left[\mathbf{o}^{(i)}(\mathbf{s}), \mathbf{m}^{(i)}(\mathbf{s})\right] \cdot \left[\mathbf{o}^{(j)(\mathbf{s})}, \mathbf{m}^{(j)}(\mathbf{s})\right] &= \mathbf{o} \cdot \mathbf{o} + \mathbf{m}^{(i)}(\mathbf{s}) \cdot \mathbf{m}^{(j)}(\mathbf{s}) \\ &\geq \|\mathbf{o}\|^2 + \frac{|\bigcap_c \mathcal{A}^{(c)}(\mathbf{s})|}{|\mathcal{A}|} \\ &> \|\mathbf{o}\|^2 = \left[\mathbf{o}^{(i)}(\mathbf{s}), \mathbf{i}^{(i)}(\mathbf{s})\right] \cdot \left[\mathbf{o}^{(j)(\mathbf{s})}, \mathbf{i}^{(j)}(\mathbf{s})\right] \end{aligned} \tag{4.9}$$

Without loss of generality, this argument of similarity improvement can be extended to cases where $\mathbf{o}^{(i)} \neq \mathbf{o}^{(j)}$, but the desired action are still the same for all agents. This fits into our environment that contains agent-specific states $\mathbf{o}_a^{(i)}$.

As such, outputs of the GAT layer will be concatenated with the transformed agent-wise observation and the current action space availability indicator:

$$\mathbf{x}^{(t,i)} = \text{MLP}_g \left(\mathbf{o}_g^{\prime(t,i)}\right) \| \text{MLP}_m \left(\mathbf{m}^{(t,i)}\right) \| \text{MLP}_{a1} \left(\mathbf{o}_a^{(t,i)}\right) \tag{4.10}$$

We use the traditional [3] and effective practice of using Long Short Term Memory (LSTM) to capture history dependency. For the current timestamp, the output feature is the hidden state $\mathbf{h}^{(t,i)}$ where:

$$\mathbf{h}^{(t,i)}, \mathbf{c}^{(t,i)} = \text{LSTM}\left(\mathbf{x}^{(t,i)}, \mathbf{h}^{(t-1,i)}, \mathbf{c}^{(t-1,i)}\right) \tag{4.11}$$

All layers up to now are reused between the actor and the critic. It has been empirically observed that parameter sharing in this way helps deep models learn at early stages [1]. We apply two final linear layers to obtain the scalar value estimate and the current action distribution, from which we sample an action to execute:

$$\begin{cases} V^{(t,i)} & = \text{MLP}_q\left(\mathbf{h}^{(t,i)}\right) \\ a^{(t,i)} & \sim \pi_\theta^{(t,i)} = \text{MLP}_{a2}\left(\mathbf{h}^{(t,i)}\right) \end{cases} \tag{4.12}$$

The method we use to obtain the policy gradient falls in the framework of advantage actor-critic [67], in which we adopt the generalized advantage estimator (GAE) [88]. At each timestamp t, the TD residual of V is defined as:

$$\delta^{(t,i)} = r^{(t,i)} + \gamma V^{(t+1,i)} - V^{(t,i)} \tag{4.13}$$

and the generalized advantage is:

$$A^{(t,i)} = \sum_{l=0}^{T-t} (\gamma\lambda)^l \delta^{(t+l,i)} \tag{4.14}$$

where T is the length of the episode and $\lambda = 1$ for unbiased estimation. The policy gradient can be thus calculated as:

$$\nabla_\theta J(\theta)^{(i)} = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta^{(t,i)} A^{(t,i)} \tag{4.15}$$

where θ denotes trainable parameters in Figure 4.1.

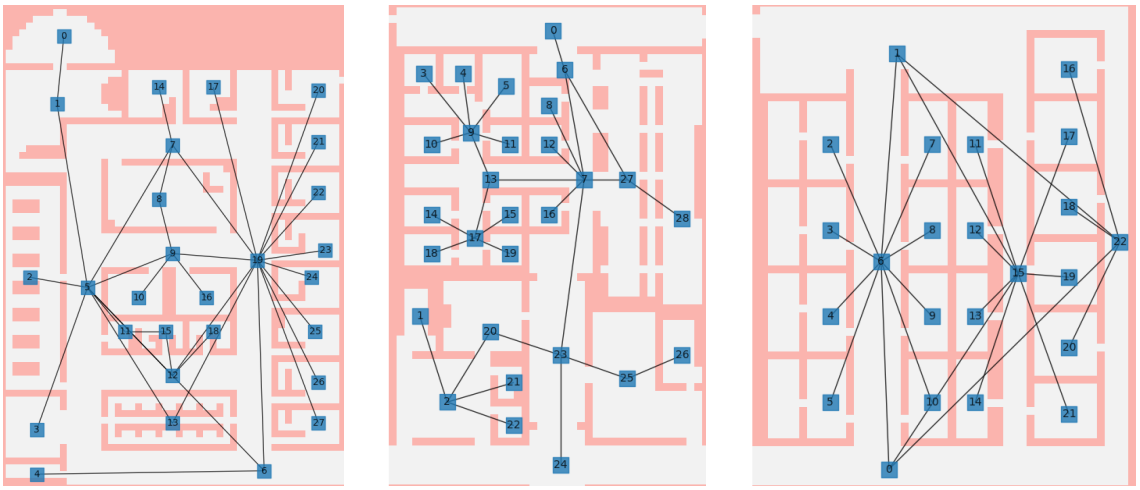
4.3 Experiments

4.3.1 Baselines

We compare the performance of our algorithm against four baselines:

- CommNet [98], where communication to agent i is the mean of hidden states from all other agents $\forall j \neq i$. The averaged message will be used as additional input channels to the agent's policy network.
- G2ANet [58], where the interaction relationship between an arbitrary agent pair i and j is modeled with a bi-directional recurrent layer following Gumbel softmax. The output will then be used as scaling factors in dot-product attention between the hidden states of i and j. Thus, their approach can be seen as applying attention over the graph of agents, where edges are weighted by the bi-directional GRU/LSTM that inputs features from corresponding agent nodes.
- QMix [83]. As discussed in Section 2.3.1, this method learns a monotonic mixer network to link agent-specific Q-value estimates with the centralized estimation. It also adopts the state-based CTDE paradigm that assumes access to true global states during training.
- Maven [63] improves QMix by introducing a latent variable z that conditions individual action-value estimators, where different z values map to different exploration traits. Global states are also required for the z-generator and the discriminator that models the variational distribution $q_v(z|\sigma(\tau))$.

As discussed above, the first two baselines rely on explicit communication. We choose them to demonstrate the benefits of utilizing attention-based communication over the environmental structure. Meanwhile, we selected the other two baselines to show that our algorithm may outperform methods that exploit training-stage global states. Moreover, all four baselines adopt the parameter sharing paradigm with plain agent-indexing; as opposed to our proposed action-dominant identification mechanism. We will benchmark our proposed approach against the baselines and conduct ablation studies to show the effect of each sub-module.



(a) Left. (b) Middle. (c) Right.

Figure 4.2: Graph-based environments.

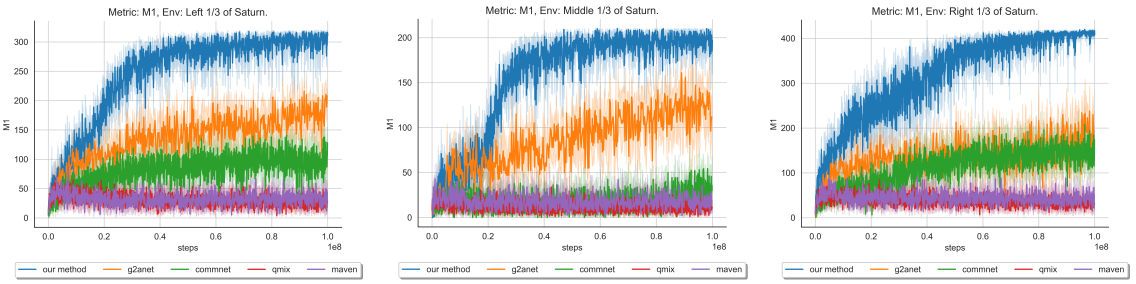
4.3.2 Results and Discussions

In this section, we quantify team performance using the final M1 score discussed in Section 3.2. We split the Saturn map in Figure 3.1 into three subdivisions with different graph structures, as shown in Figure 4.2:

- The left region with 28 nodes (12 of which are blocked by a total of 19 rubbles) and 33 edges. This environment has 12 victims, resulting a total M1 of 320.
- The middle region with 29 nodes (13 of which are blocked by a total of 22 rubbles) and 29 edges. This environment is the most sparse in terms of victim distribution, containing 9 victims and a total M1 of 210.
- The right region with 23 nodes (13 of which are blocked by a total of 15 rubbles) and 24 edges. 14 victims are trapped in this environment, including a room with three critical victims worth 150 M1 scores. The total M1 available in this environment is 420.

The same discount factor $\gamma = 0.7$ is used in all environments. Each episode contains 400 steps, and room contents are randomly permuted across episodes. In addition, shades in the following result plots indicate the 95% confidence interval.

Figure 4.3 demonstrates that our method outperformed baselines in all three map settings. Our model performed particularly well in the right-most region that bias



(a) Left. (b) Middle. (c) Right.

Figure 4.3: Performance in the marcomanagement task.

toward role-specific interactive actions, achieving a 2.2× boost.

We observed that MAVEN, which has a theoretical advantage over QMix, outperformed the latter approach in all scenarios. However, they still performed significantly worse than communication-based architectures. The main reason behind this phenomenon is not insufficient replay buffer size or poor choices of other hyperparameters, but the fact that observations are highly sparse. We demonstrate so by providing global states to MAVEN as additional inputs during both training and evacuation. The model architectures remain the same except for the number of input channels. As shown in Figure 4.4, the (mostly) identical architecture and hyperparameter choices could effectively learn a good mapping from $(\mathbf{s}, \mathbf{o})$ to $\mathbf{a}$, but were unable to address the strong partial observability in USAR tasks. Thus, we argue that explicit communication schemes should be applied when the task setup allows execution-stage information exchange.

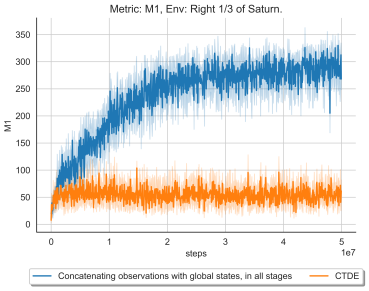
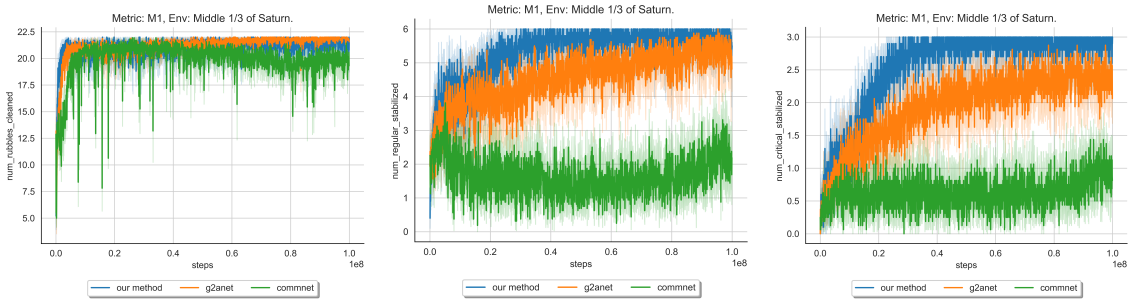


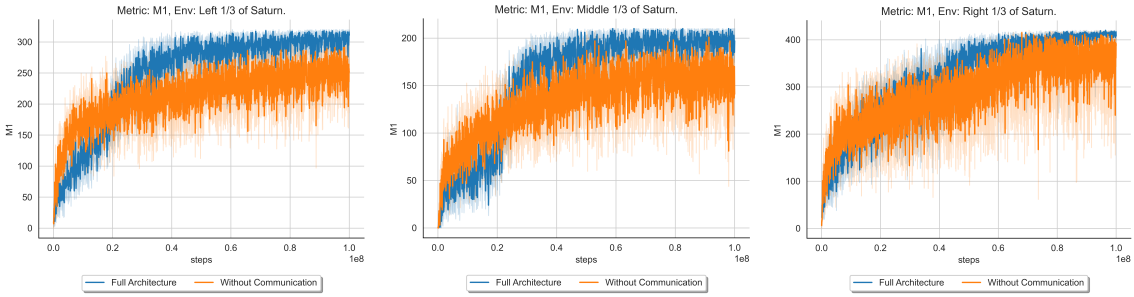
Figure 4.4: Impact of partial observability on MAVEN.

We also notice that G2ANet outperformed CommNet in all environments, and the difference is most significant in the middle one. As successful victim evacuation relies on multiple preconditions, we plot the performance of individual tasks in Figure 4.5. All three algorithms were able to clean almost all rubbles, as shown in Figure 4.5a. However, CommNet failed to effectively accomplish secondary tasks that require more team collaboration. Figure 4.5c shows how the ability to stabilize critical victims is strongly correlated with the eventual task performance.



(a) Rubble Cleaning. (b) Regular Victim Stabilization. (c) Critical Victim Stabilization.

Figure 4.5: Performance in preconditions - Middle Map.



(a) Left. (b) Middle. (c) Right.

Figure 4.6: Ablation studies - Communication

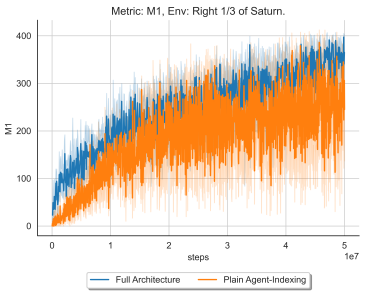


Figure 4.7: Ablation studies: Agent-indexing schemes.

Moreover, we conduct ablation studies regarding the communication mechanism to see how it contributes to the overall excellence of the proposed architecture. As shown in Figure 4.6, communication between agents reduced variance because the learner now perceives a more stationary environment. This effect is more significant in the left and middle maps, leading to not only variance reduction but also more optimal convergence. This is because their underlying graph structures are more complex and victims are more sparsely distributed. Meanwhile, Figure 4.7 shows how information regarding an agent's action space can be used as a strong indicator of the essence of that agent, which greatly stabilized training.

We also show how the attention mechanism helps coordination. Let us revisit the notation in Section 4.2 where p denotes an arbitrary node and $q \in Q_p \doteq \mathcal{N}(p) \cup \{p\}$ belongs to its neighborhood; $W \in \mathbb{R}^{f_g \times f_g}$ denotes the linear weights that encodes graph features, and $\mathbf{a} = [\mathbf{a}_P; \mathbf{a}_Q] \in \mathbb{R}^{2f_g \times 1}$ is the attention layer's weights shared for all pairs (p, q). From Equation 4.5, we can construct the vectors:

$$\begin{aligned} \mathbf{w}_P &= W^\top \mathbf{a}_P \in \mathbb{R}^{f_g \times 1} \\ \mathbf{w}_Q &= W^\top \mathbf{a}_Q \in \mathbb{R}^{f_g \times 1} \end{aligned} \tag{4.16}$$

such that the raw attention coefficients $e_{p,q}$ indicating the importance of node q's contents to the current node p can be computed as:

$$e_{p,q} = \text{LeakyReLU}\left(\mathbf{w}_P^\top \bar{\mathbf{o}}_g(p) + \mathbf{w}_Q^\top \bar{\mathbf{o}}_g(q)\right), \forall q \in Q_p \tag{4.17}$$

Here, $\mathbf{w}_Q$ explains how any node values its neighbors by feature dimensions. Meanwhile, note that $\mathbf{w}_P^\top \bar{\mathbf{o}}_g(p)$ only encodes contents of the current node p, thus serves as a bias term uniform to all neighbors of p. Because of the structure of the LeakyReLU activation, $\mathbf{w}_P$ declares what node p should contain so that the attention mechanism will examine the neighborhood of p more **precisely**. Given $\mathbf{w}_Q \bar{\mathbf{o}}_g(q)$, if the feature distribution of node p has a (sufficiently) higher similarity with $\mathbf{w}_P$, the resulting greater dot-product $\mathbf{w}_P^\top \bar{\mathbf{o}}_g(p)$ will cause more $z_q \doteq \mathbf{w}_P^\top \bar{\mathbf{o}}_g(p) + \mathbf{w}_Q^\top \bar{\mathbf{o}}_g(q)$ to be greater than 0. Because more z_q values will be activated (by a unit tangential $1 > \alpha$), the pairwise distances between $e_{p,q}$ will be greater, leading to greater pairwise distances between $\alpha_{p,q}$ and closer discrimination upon the neighborhood. Also, note that higher overall $e_{p,q}$ caused by an increase $\mathbf{w}_P^\top \bar{\mathbf{o}}_g(p)$ would not cause the network to put uniformly more attention to all neighbors, as the actual attention coefficients $\alpha_{p,q}$ are subject to the SoftMax normalization with translation invariance.

We plot $\mathbf{w}_P, \mathbf{w}_Q$, each independently normalized, with feature labels in Figure 4.8. It is immediately observed that the network cares to closely examine the neighborhood only when unstabilized critical victims are present in the current room p. This phenomenon matches our intuition, as stabilizing a critical victim is the only atomic action that demands collaboration (Section 3.2). The network, upon seeing an unstabilized critical victim, will try to search for agents in adjacent rooms and instruct them to come and help. This is also reflected in $\mathbf{w}_Q$, as the model cares

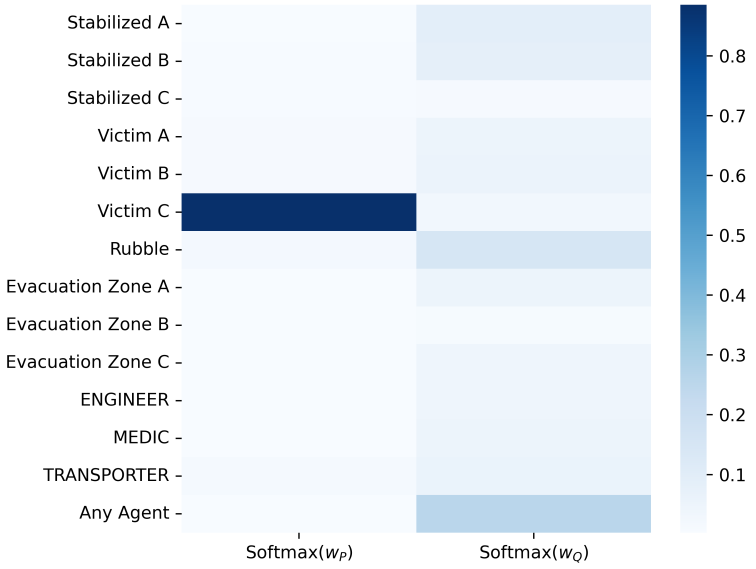


Figure 4.8: Attention to feature dimensions. Weights are extracted from the model trained for 100 million steps in the Saturn-right map.

whether an arbitrary agent resides in the neighboring rooms. In addition, we observed that rubbles in the neighboring rooms are also closely attended to. We hypothesize that this is because rubble cleaning is a task valuable in both the short-term (easy rewards that can be achieved at the cost of only one agent-step) and the long run (cleaning is essential to allow for future task execution).

Chapter 5

Micromanagement

5.1 Grid-based Environment Simulator

For the task of micromanagement, we developed a testbed[1] that provides observation spaces with a 1m $\times$ 1m resolution and action spaces transferable across participants.

5.1.1 State and Observation Space

The global state $s \in \mathcal{S}$ and agent-role mapping C are represented together with an array in shape `(ROW, COL, 6)`. The observation $\mathbf{o}^{(t,i)} \in \Omega$ is filtered from $\mathcal{S} \cup \mathcal{C}$ by a Boolean visibility mask with a 51° field of view. Walls and rubbles that are at least two meters tall would cause occlusion, as shown in Figure 5.1. For RL-based agents, they are normalized to $[0, 1]$ channel-wise, whereas planning-based agents perceive the raw `uint8` encoding.

5.1.2 Action Space and Dynamics

The action space $\mathcal{A}$, which contains `still`, `forward_1`, `forward_2`, `forward_3`, `turn_left`, `turn_right`, `toggle`, `pick_up`, is uniform to all agents. The purposes of the first six navigational actions are self-explanatory. In addition, all agents in the grid-based environment may pick up a stabilized victim and carry it to the

[1] https://gitlab.com/cmu_asist/multigrid-with-3-roles

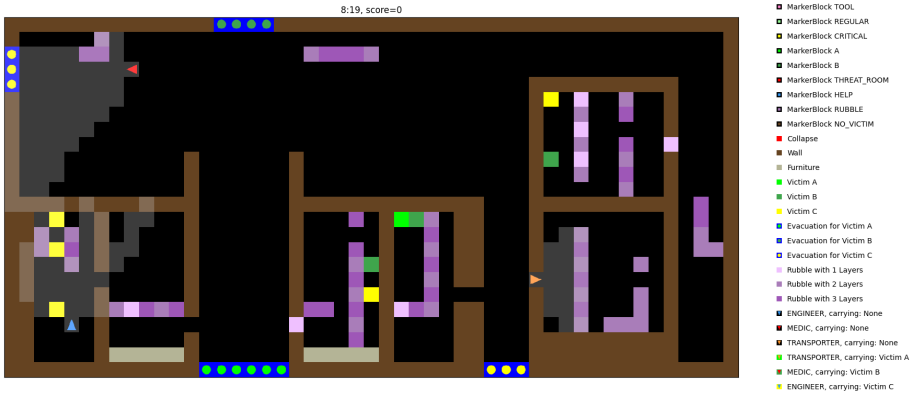


Figure 5.1: Visibility masks that overlay the global map. Light Grey indicate that the block is within an agent's field of view.

evacuation zone; or relocate a regular victim for the convenience of other agents. Heterogeneity regarding evacuation is modeled with role-specific speeds described in Table 5.1. Furthermore, in order to decouple heterogeneous behavior from the action space, every role has the permission to call for any of the `forward` actions. The environment will ensure that the expected value of an agent's speed is consistent with its corresponding fractional speed limit.

Role	Carrying a victim	Not carrying a victim
Medic	2.374	4.317
Engineer	2.374	3.669
Transporter	5.180	5.180

Table 5.1: Traveling speed [m/s] of each role. The environment operates at 4Hz.

The `toggle` action is used as the main interaction method with the facing block, and should be called for a medic to stabilize an injured victim, for an engineer to clean a rubble block, or for a transporter to evacuate a stabilized victim being carried. If an agent with a role incompatible with the current facing block calls `toggle`, the environment will ignore its behavior or optionally return a small negative reward. This deliberately ambiguous `toggle` action lead to role-specific dynamics:

$$\forall 1 \leq c_1, c_2 \leq C : c_1 \neq c_2, \exists\, s, s' \in \mathcal{S} : T^{(c_1)}(s, \text{toggle}, s') \neq T^{(c_2)}(s, \text{toggle}, s') \quad (5.1)$$

Meanwhile, such ambiguity can be handled by the environment without collision.

A separate action, `pick_up`, is introduced in case a medic hopes to move a regular victim that has yet been stabilized. In addition, note that the reward structure stays homogeneous over roles, given any tuple (s, a, s').

5.1.3 Optional Features

The grid-based simulator has the capability to mimic the 3D Minecraft testbed for human trials (Appendix A) to the fullest extent possible. These optional features are only used for planning-based agents. Below is a non-comprehensive list:

- Enlarged Action Space: A `swap` action is introduced to address the issue with a discretized 2D testbed, which can be called when a participant, who is already carrying a victim, finds its path blocked by a movable victim. The action would swap the location of this participant with the movable victim, which can be done in a few steps in the 3D testbed. Also, a `forced_drop` action can be called when a participant chooses to drop a victim at the evacuation location, with no regard for the injury types. On the contrary, calling `toggle` when facing a wrong-typed evacuation block would prevent the participant from dropping the victim, as if the participant noticed the no-change in the scoreboard in the 3D testbed.
- Collapse during the mission: as the building structure is unstable following the earthquake, agents may accidentally trigger secondary collapse while attempting the rescue. When enabled, new rubbles would fall and block the agent's path out, trapping it until the engineer came to help. This calls for closer collaboration between team members.
- Task-based visibility mask: non-medic participants cannot differentiate between abrasion and bone damage victims, and they will have no access to any injury types after a victim has been stabilized. Moreover, collapse traps (not common rubbles) cannot be seen by any participants once the task has started, although the engineer has room-level prior knowledge regarding these traps.
- Marker blocks: to address the constraint discussed above, a medic may place marker blocks indicating injury types. These markers are globally visible (independent of the visibility masks), in order to mimic the dynamic map in

the 3D testbed.

- Beep signals: the transporter may request for extra information when stepping on a trigger block. The environment would then return whether non-stabilized regular or critical victims are in the corresponding room.

5.2 Algorithms

For decision-making in the micromanagement scenario, we first adopt the paradigm of online reinforcement learning, like what we did in Chapter 4. We use a CNN shared between the actor and the critic to extract features from image-like observations. We then feed the features to two MLP branches separated for the value function and the actor head. The entire network is shared among all agents with no additional regard to their identity. Furthermore, we use the same mechanism discussed in Section 4.2 to estimate advantages and calculate the policy gradient.

In order to use imitation learning for our USAR mission, we develop a hierarchical controller equipped with structured communication protocols [72]. A priority queue is maintained to host communication instances. Agents may request each other to gather, clean rubbles, move victims, query for information, or respond to previous requests. An agent, upon receiving an action-related request, will compare it with the current task in-execution and decide accordingly. Meanwhile, an agent at ease would search in its internal belief system for the most valuable room-level action with exponential exploration. The room-level action would be translated to a Boolean mask of target grid-level locations, from which A* planning [38] is applied for navigation. In addition, in order to address the non-stationarity encountered with independent execution with insufficient knowledge sharing, agents will have an increasing tendency to revisit states that deemed invaluable according to its memory. Figure 5.2 shows the structure of the proposed planning-based agents.

The non-Markov experts that rely on a shared communication queue can be modeled with a joint policy $b(A|O, \mathbf{h})$, where $O \in \prod_i \Omega^{C(i)}$, $A \in \prod_i \mathcal{A}^{C(i)}$ are joint observations and actions. We use the expert policy to roll out episodes with observation-action pairs $\left\{(O^{(t)}, A^{(t)})\right\}$, and decompose them back to the individual representation of $\left\{(\mathbf{o}^{(t,i)}, \mathbf{a}^{(t,i)})\right\}$. Using the expert trajectories as ground truth, we apply supervised

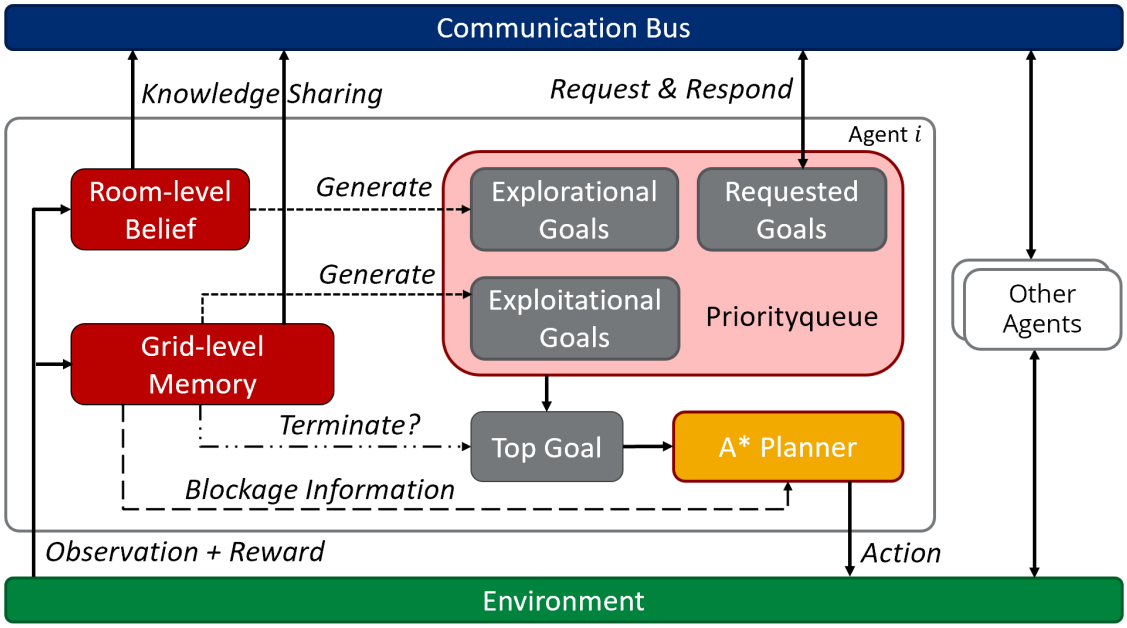


Figure 5.2: Hierarchical planning agents.

learning to optimize for the IL-agent's policy network:

$$\min_{\theta} -\sum_{i,t} \log \pi_\theta \left(\mathbf{a}^{(t,i)} | \mathbf{o}^{(t,i)}\right) \tag{5.2}$$

where π_θ has an identical network architecture as the actor used in online RL, for the sake of a fair comparison.

We also consider the goal of forecasting human performance, in terms of the M1 metric introduced in Section 3.2, in two time scales. Let $m^{(t)}$ denote the cumulative M1 score at timestamp t. Given the sequence $\{m^{(0)}, ..., m^{(t)}\}$, we are to infer either the short-term progression $m^{(t+\Delta t)}$ or the eventual outcome $m^{(T)}$. We address this sequence modeling problem with a transformer encoder [105] exploiting multi-head self-attention. The transformer is fed with aggregated features of the trajectory to reduce the demand for computational resources. Moreover, we consider a baseline step-wise MLP model that does not consider time dependency. We use this baseline to evaluate the necessity of using the data-intensive transformer architecture.

5.3 Experiments

5.3.1 Decision-Making

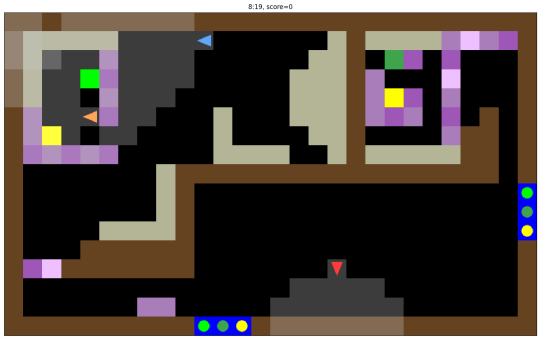
Experiment Design

The grid-based environment simulator supports multiple randomization levels. More stochasticity, *e.g.*, to permute the location of all elements except for the building structure, leads to a higher requirement for the expressivity of the network. Meanwhile, a less randomized environment may require the engineer to clean rubbles prior to victim stabilization, posing a stricter requirement for team collaboration. In this section, we select four representative configurations, as shown in Figure 5.3, and see how randomization levels affect the performance of RL and IL models. We generated 2000 expert trajectories for each of these environment configurations.

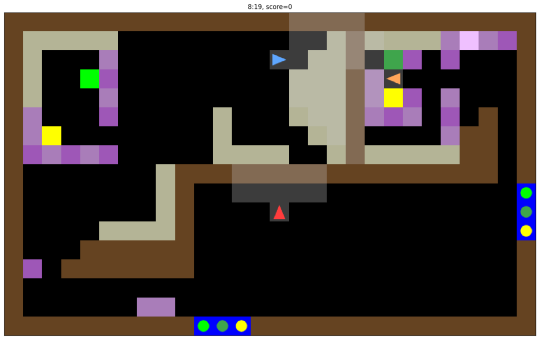
Results and Discussions

Figure 5.4a shows how the online RL paradigm learns to improve episodic rewards. However, such improvements were mainly contributed by the engineer's increasing ability to clean rubbles, as shown in Figure 5.4b. As rubble-cleaning is a task that can be accomplished by the engineer alone without any pre-conditions, limited team collaboration was achieved in the online RL setup. This is reflected in Figure 5.4c and 5.4d. The shared policy never learned to stabilize victims exceeding the extent of random walking, let alone to evacuate them afterward. One explanation for this phenomenon is action space ambiguity. The `toggle` action presented to the policy network corresponds to two unbalanced underlying tasks: rubble cleaning and victim stabilization. As the number of rubbles greatly exceeds the number of victims, the shared policy network perceived the sparse occasions of victim stabilization as noise, thus failing to optimize in that direction meaningfully. A potential solution within online RL is carefully handcrafting the reward structure and reducing the rewards associated with rubble cleaning.

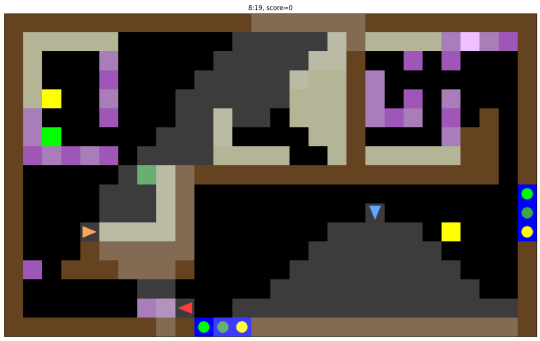
Figure 5.5 demonstrates the benefits of imitating faux-human trajectories over purely online RL. It is shown that IL managed to accomplish the entire decision sequence of cleaning-stabilizing-evacuation, despite the same network architecture



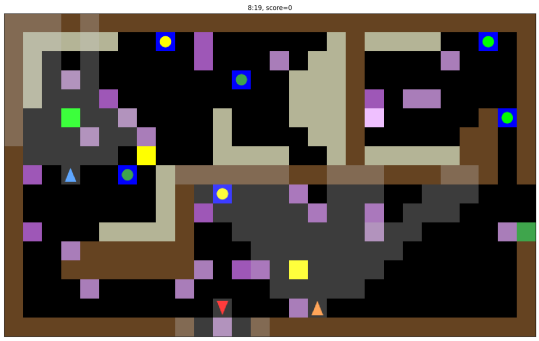
(a) Fixed environment with a hard collaboration requirement.



(b) Fixed environment with a soft collaboration requirement.



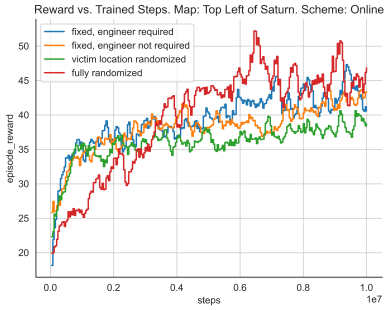
(c) Victim locations are randomized.



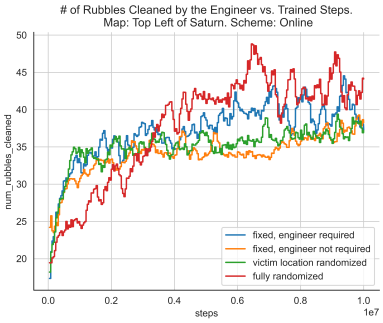
(d) All element locations are randomized.

Figure 5.3: Four randomization levels in the top-left region of Saturn. The initial agent locations are always randomized. Environment dimension: $17 \times 28 \times 6$. Maximum M1: 120.

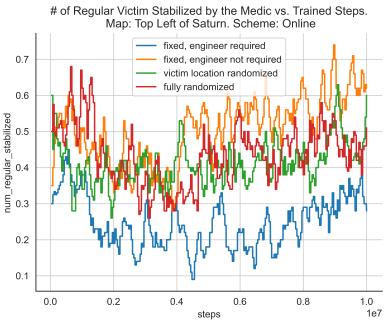
being used by both learning schemes. Meanwhile, the agents' performance was subject to environment configurations. As shown in Figure 5.5b, posing a hard constraint on team collaboration would decrease the cumulative M1 metric by a minor margin. Regarding the topic of randomization, it is shown that agents were able to mitigate challenges associated with randomized victim locations. However, when the locations of rubbles and evacuation zones were permuted across episodes as well, a significant performance reduction was observed. This could be explained by two hypotheses. Either a) the representation power of the policy network or b) the size and variance of the expert dataset were insufficient for the more sparsely distributed observation space. As issues caused by insufficient expert demonstrations and distribution shifts



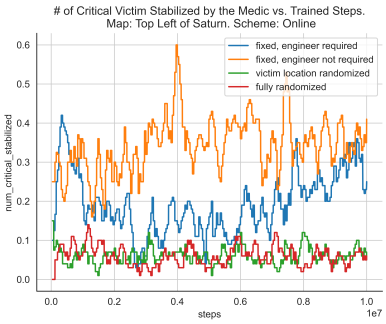
(a) Rewards received.



(b) Rubbles cleaned.

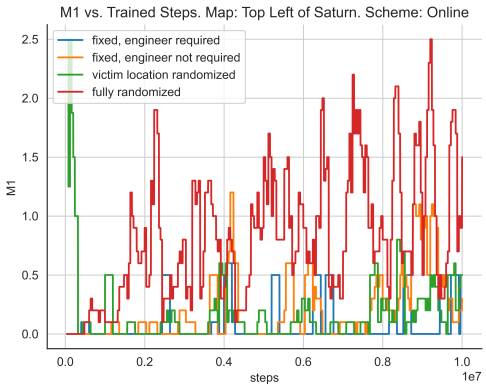


(c) Regular victims stabilized.

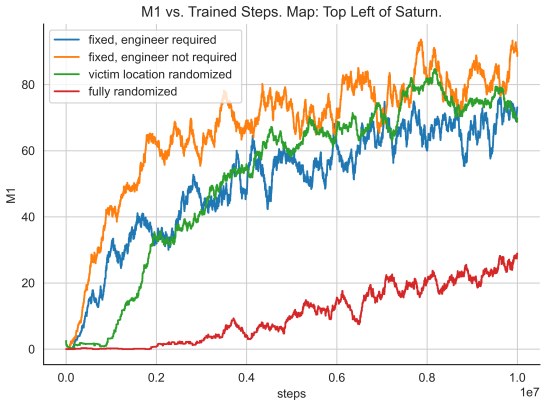


(d) Critical victims stabilized.

Figure 5.4: Details regarding online RL.



(a) Online RL.



(b) IL.

Figure 5.5: Comparison between online RL and IL over end-of-episode performance, in the environment shown in Figure 5.3.

were commonly observed in behavior cloning applications, the latter hypothesis is more likely to have played a more significant role.

5.3.2 Forecasting Human Performance

Experiment Design

In this section, we use the full-sized Saturn Map as shown in Figure 3.1 with optional features in Section 5.1.3 enabled. The duration of the mission is 15 minutes, which translates to 3600 steps in the environment. Our collaborators [19] first conducted 12 human trials in an equivalent environment, which will be referred to as D_h. We use the observations and statistics from D_h to configure our faux-human agents, and roll out 3000 episodes denoted as D_f. After the faux-human dataset was collected, our collaborators collected 32 human trials, denoted as D_{vh}, with the same condition as D_h. They collected more 184 human trials, denoted as D_{vi}, where participants receive external guidance. Thus, D_h and D_{vh} should follow a similar distribution, where D_h and D_{vi} are likely to follow different distributions. The rationale of such a data split is discussed in Appendix A.

Under these preparations, we consider two experimental designs. Firstly, we study the effect of adopting synthetic data at an early stage of design iterations. For this purpose, we train the prediction models with either D_h or D_f, and evaluate the models with D_{vh} that is made available after the training sets. Secondly, we investigate whether synthetic data effectively span the policy space, such that they can be used to bridge the gap in human data caused by assistive robots or human advisors. In this case, we train the predictors with either D_{vh} or $D_{vh} \cup D_f$, and evaluate the models with D_{vi}. Furthermore, each of these two settings involves both the short-term ($\Delta t = 60$s) and long-term prediction tasks.

To measure the performance of prediction models, we report the root mean square error (RMSE) between prediction and ground truth as the y-axis in the following figures. Because the synthetic dataset is significantly larger in size than human datasets, plotting errors against trained epochs would exaggerate the advantages of using synthetic data. Thus, we trained the models concurrently on the same devices and use training time as the x-axis.

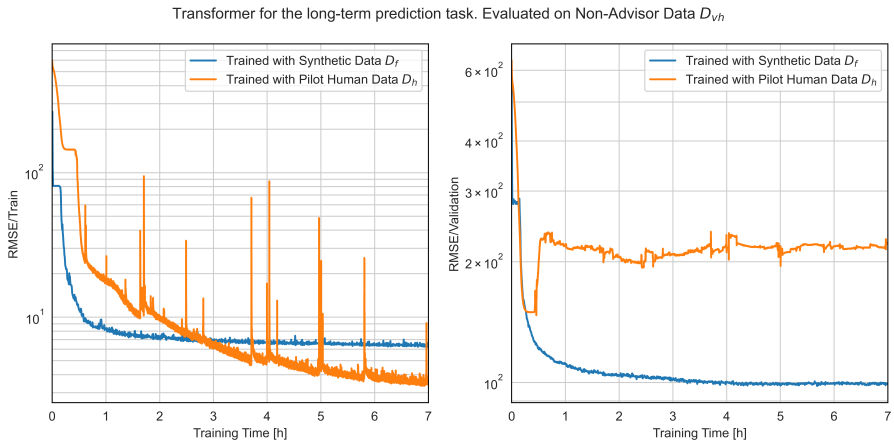


Figure 5.6: Long-term prediction model: $D_h|D_f \rightarrow D_{vh}$.

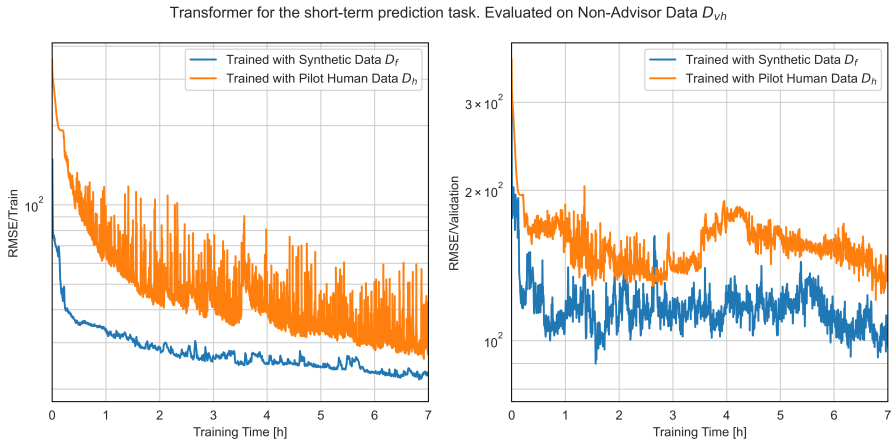


Figure 5.7: Short-term prediction model: $D_h|D_f \rightarrow D_{vh}$.

Results and Discussions

Figure 5.6 and 5.7 compare training with synthetic data vs. training with limited human data collected for early design choices. Figure 5.8 and 5.9 demonstrate the impact of augmenting the non-advisor human datasets with synthetic data. While introducing the faux-human dataset reduced validation error in all four configurations, we observed that performance improvement is more significant in long-term prediction tasks. This is because long-term prediction is modeled as a (simpler) sequence-to-one problem, leading to a higher chance of overfitting. As shown in Figure 5.6, when trained on the limited human dataset D_h, the model would quickly overfit after 30

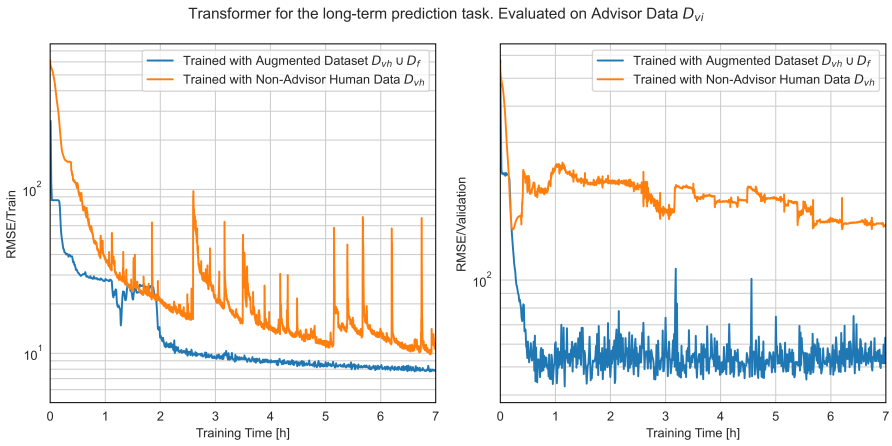


Figure 5.8: Long-term prediction model: $D_{vh}|D_f \cup D_{vh} \rightarrow D_{vi}$.

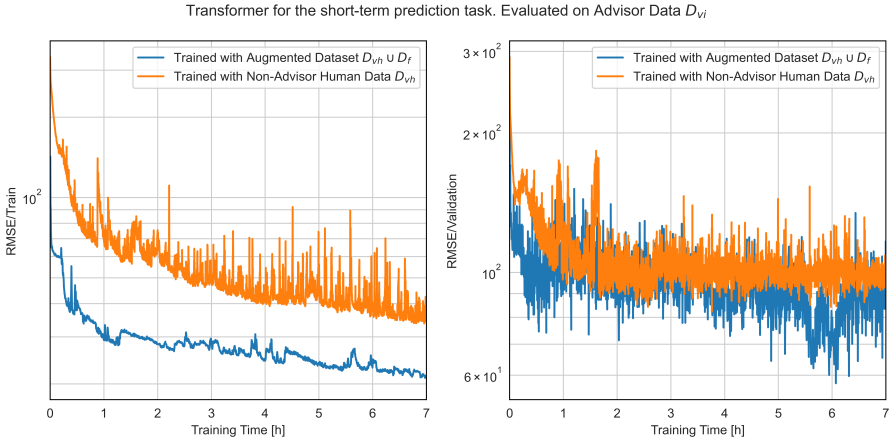


Figure 5.9: Short-term prediction model: $D_{vh}|D_f \cup D_{vh} \rightarrow D_{vi}$.

minutes of training. Meanwhile, as the tendency of overfitting is insignificant in the sequence-to-sequence problem of short-term prediction, the main benefit of adopting the synthetic dataset is that it helps stabilizes the transformer. Figure 5.7 and 5.9 show how training with human data only leads to a higher variance in training RMSE.

Figure 5.10 presents the comparison between the transformer and the baseline MLP model, where the latter one holds 65% less trainable parameters. Whether synthetic data is used or not, it is shown that the ability to consider time dependency is vital for the long-term task, resulting in lower training and validation RMSE. This invalidates the counterargument that relatively worse performance from training on

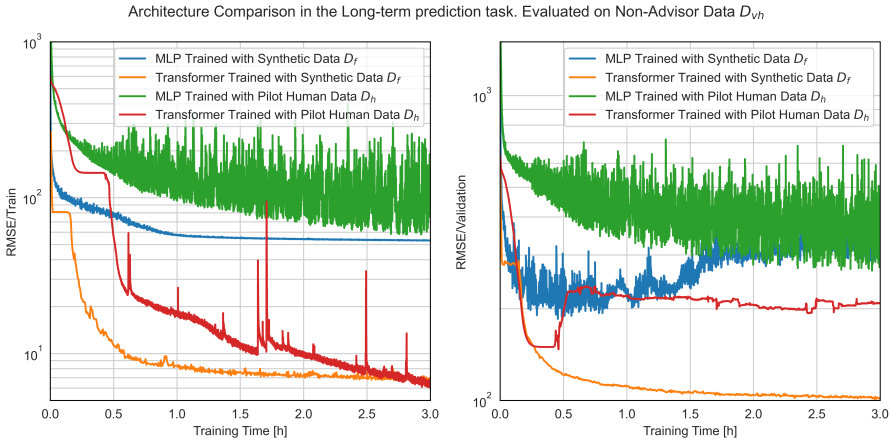


Figure 5.10: Comparison between MLP and Transformer for long-term prediction.

human data alone can be addressed by plainly switching to a simpler model. Thus, we argue that applying data-intensive sequential models together with synthetic data is an adequate approach for human performance forecasting.

Chapter 6

Conclusion

In this thesis, we studied the task of coordinating a heterogeneous team in urban search and rescue (USAR) missions. For this purpose, we developed two USAR environment simulators that support role-specific action spaces or transition dynamics. We applied the graph attention mechanism to exploit environment structures and achieve effective communication among a small group of rescue workers. We showed that the proposed approach outperformed previous state-of-the-art literature that also rely on graph attention by a significant margin. In addition, we designed a series of faux-human agents that vividly mimic human behavior in USAR. We demonstrated how they could be used as experts in imitation learning to address action-space ambiguity. In the end, we showed that our artificial datasets addressed human data scarcity. We reduced the inference error of human cognitive models by half via this method of data augmentation.

Appendix A

Acquiring Human Data

Our collaborators [19, 29] developed a 3D synthetic environment for USAR using Minecraft. Figure A.1 shows the bird's eye view of the environment, whose map structure and transition dynamics are equivalent to those of our 2D grid-based simulator in Figure 3.1.

Figure A.1: Minecraft Environment

Participants are recruited from Arizona State University and social platforms, *e.g.*, Reddit. They are physically located in the US, have reliable internet connections,

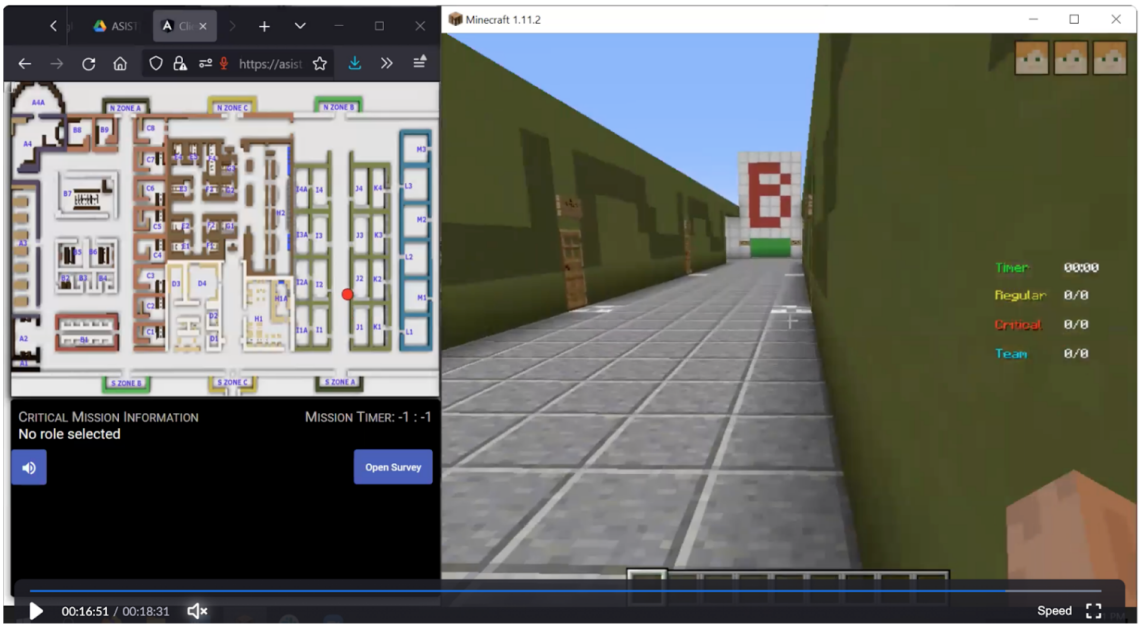


Figure A.2: The layout of a participant's interface.

have experience playing Minecraft with a standalone mouse and keyboard, speak English, and have normal color vision. The recruitment excludes minors and prisoners. Furthermore, participants are asked to team with people who they do not know in advance, so as to control the variance in collaboration level.

Prior to a human trial, participants are subject to a 10-minute screening session to check their capability to play Minecraft. Following the screening session, teams of three qualified participants will receive training videos that introduce the rules of the task and provide hands-on experience with the environment. Each team will then participate in two seventeen-minute trials in the Minecraft simulator, where participants use the first two minutes to discuss strategy and the following fifteen minutes to execute the mission. Each participant perceives the partially observable interface as shown in Figure A.2. The right 60% of the screen shows the corresponding agent's field of view, whereas the left 40% of the screen reflects the map structure of the environment.

A participant sees their own location on the dynamic map, but not the location of their teammates. To mitigate this visibility constraint and share other information, a participant may place marker blocks on the ground, which are also rendered in

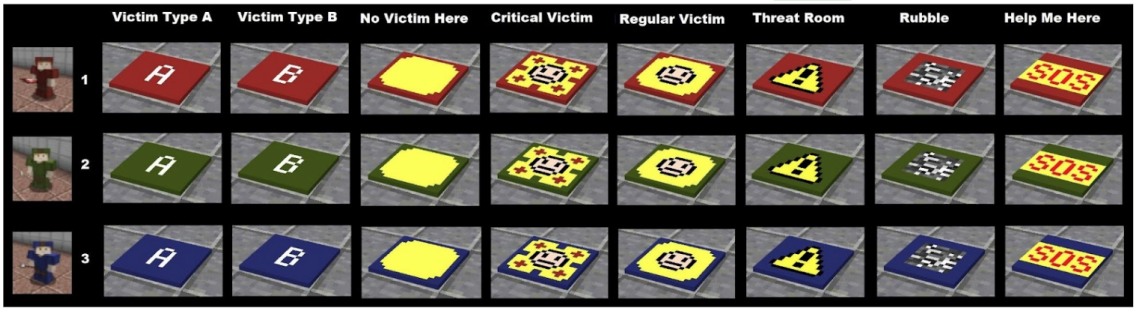


Figure A.3: Marker Blocks

all teammates' dynamic maps. These marker blocks can be used to indicate the injury type of a victim nearby, that the room is blocked by rubbles, that the room contains collapse threats, or if the room is empty. As shown in Figure A.3, they also reflect the identity of the marker-placer, where blue corresponds to Engineer, green corresponds to Transporter, and red corresponds to Medic. In addition to the marker blocks, participants may verbally communicate over zoom audio and use Minecraft chat channels.

One purpose of acquiring human trajectories is to support the development of assistive robots that issue reasonable interventions to improve human teamwork. Thus, as discussed in Section 5.3.2, human data are collected under different conditions:

- D_h, which includes 12 trials collected at a pilot stage. This dataset is used to guide the early development of assistive robots (intervention modules).
- D_{vi}, which includes 184 trials collected after the development of assistive robots. The robotic advisors are deployed into the environment during these trials, where they issue interventions to human participants as they see fit.
- D_{vh}, which includes 32 trials also collected after the development of assistive robots. However, the robots are deactivated during these trials, thus D_{vh} serves as the comparison group.

Appendix B

Implementation and Future Work

The Adam optimizer [50] is used to train all our models. In addition, the random seeds for the environment and MARL/MAIL/Inference algorithms are selected randomly from OS-provided system entropy, so as to avoid cherry-picked results.

B.1 Marcomanagement

The network architecture[1] and hyperparameters used to obtain the main results are shown in Table B.1.

Optimization & Env.	**Value**	**Network Architecture**	**Value**
Discount Factor γ	0.7	Attention Layers	2
Learning Rate	10^{-4}	Attention Heads	1
Batch Size	200	LeakyReLU α	0.2
Graph Channels F_g	14	Attention Output F'_g	14
Agent Channels F_a	6	LSTM Channels $F_h = F_c$	256

Table B.1: Hyperparameters for marcomanagement.

Moreover, the feature-embedding MLPs in Figure 4.1 following $\mathbf{o}'_g, \mathbf{o}_a, \mathbf{m}$ contain two consecutive linear layers with 256 output channels each. The value branch to estimate V is a linear layer (which has one output channel), and the policy branch to calculate the distribution of $\pi(\cdot|\sim)$ is also a single linear layer.

[1]Implemented at https://gitlab.com/cmu_asist/MARL-Heterogeneous

We conducted hyperparameter searches for $\gamma \in [0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99]$, the learning rate in $[10^{-3}, 10^{-4}, 10^{-5}]$, the training batch size in $[20, 200, 2000]$, the number of attention layers in $[1, 2, 3]$, and the number of attention heads in $[1, 2, 4, 8, 16]$. We observed that changing the order of the learning rate and batch size would cause the model to diverge or converge to local maxima. Among the rest hyperparameters, we observed that the discount factor γ was the most important. Thus, we use the same discount factor $\gamma = 0.7$ for our baseline methods[2] for a fair comparison.

Results in Chapter 4 are reported based on 10 evaluation episodes at each plotted point. The dense curves indicate the mean of evaluation episodes, whereas the shades represent the 95-th confidence interval bootstrapped from 20 up-samples.

B.2 Micromanagement

The hyperparameters used to obtain the main results are shown in Table B.2.

Optimization & Env.	Value
Discount Factor γ	0.7
Learning Rate	10^{-4}
Batch Size	200
Grid Width `ROW`	28
Grid Height `COL`	17
Input Channels	6

Table B.2: Hyperparameters for micromanagement.

The network[3] contains four consecutive convolutional layers (with ReLU activations in between). The first one has a 3×3 filter and 16 output channels; the second one also has a 3×3 filter and 32 output channels; and the third one uses a filter, which has the same size as the input 2D grid, to map the image-like features to a $1 \times 1 \times 32$ output. The output is then fed into the final convolutional layer with a 1×1 filter to obtain the action probability distribution. As mentioned in Chapter 5, this architecture is used across both online RL and imitation learning.

[2]The baseline methods were originally implemented by https://github.com/starry-sky6688/MARL-Algorithms. We applied necessary adaptions and bug fixes at our forked repository https://github.com/andromeda-0/StarCraft/releases/tag/v1.0.

[3]Implemented at https://gitlab.com/cmu_asist/MAIL-Heterogeneous

We put careful consideration into the faux-human trajectories' reproducibility. For each trial, a parent seed sequence is generated from OS-provided system entropy, then spawned into two child seed sequences to initialize the independent random number generators used by the grid-world environment and agents, respectively. This practice is known to avoid seed collision [37]. These seed sequences are saved together with environment and agent configurations, which fully suffice to reproduce any trial.

B.3 Predicting Human Performance

Both human data and synthetic data are parsed to time sequences with a frequency of $4Hz$. We consider the following input features at every timestamp:

- `positions` is a 3×2 array indicating the $x - y$ location of each agent.
- `ratio_unvisited` is the ratio of rooms that are yet visited by any agent.
- `time_in_seconds` is self-explanatory.
- `<A∪B|C>_<discovered|moved|stabilized|signaled|evacuated>` contains ten integers that indicate the task progress. Note that victim discovery is defined as the union of victim stabilization and signaling.

The network architecture[4] is as follows: the input features are flattened, concatenated, and fed into a linear layer with 64 output channels. The output, activated by a ReLU layer, is fed into a transformer encoder with dual-head attention and 256 output channels. In the end, the 256-dim vector is mapped to a scalar prediction with a linear layer. Meanwhile, the baseline MLP model replaced the transformer encoder with a linear layer (that has the same output channels) and a ReLU activation function.

B.4 Future Work

We acknowledge two aspects of future work:

- To outperform planning-based experts in the micromanagement task. A potential solution is to consider IL or offline RL as pre-training, where the models

[4]Implemented at https://gitlab.com/cmu_asist/Inference_Artificial_Trajectories

trained from offline demonstrations are further fine-tuned with online interactions with the environment [30, 33, 82].

- To investigate the generalization capability in the macromanagement task. The graphs on which we experimented in Chapter 4 are sparse ($E \approx G$) and contain at most thirty nodes. Can our proposed architecture demonstrate a similarly significant improvement over the baselines, in a larger or denser graph? In order to do so, we may need to increase the model capacity, *e.g.*, to use more attention heads to attend to different features in neighboring nodes. However, preliminary results suggested that the benefit of using plain dot-product multi-head attention is limited in our setup, despite its success in various domains [24, 105]. A potential solution is to regularize the model for higher diversity and sparsity among the attention heads [20, 113, 124].

Bibliography

[1] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, and others. What matters in on-policy reinforcement learning? a large-scale empirical study. *arXiv preprint arXiv:2006.05990*, 2020. 4.2

[2] Itamar Arel, Cong Liu, Tom Urbanik, and Airton G. Kohls. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128–135, 2010. Publisher: IET. 2.2.2

[3] Bram Bakker. Reinforcement Learning with Long Short-Term Memory. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper/2001/hash/a38b16173474ba8b1a95bcbc30d3b8a5-Abstract.html. 4.2

[4] Thomas Barrett, William Clements, Jakob Foerster, and Alex Lvovsky. Exploratory Combinatorial Optimization with Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3243–3250, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i04.5723. URL https://ojs.aaai.org/index.php/AAAI/article/view/5723. Number: 04. 2.4

[5] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. 2.3.1

[6] Richard Bellman. On the Theory of Dynamic Programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716–719, August 1952. ISSN 0027-8424. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1063639/. 2.1

[7] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002. Publisher: INFORMS. 2.2.1, 2.2.2

[8] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *TARK*, volume 96, pages 195–210. Citeseer, 1996. 2.2.1

[9] Dong Chen, Zhaojian Li, Yongqiang Wang, Longsheng Jiang, and Yue Wang.

Deep multi-agent reinforcement learning for highway on-ramp merging in mixed traffic. *arXiv preprint arXiv:2105.05701*, 2021. 1

[10] Haoqiang Chen, Yadong Liu, Zongtan Zhou, Dewen Hu, and Ming Zhang. GAMA: Graph Attention Multi-agent reinforcement learning algorithm for cooperation. *Applied Intelligence*, 50(12):4195–4205, December 2020. ISSN 1573-7497. doi: 10.1007/s10489-020-01755-8. URL https://doi.org/10.1007/s10489-020-01755-8. 1, 2.3.2

[11] Tianyi Chen, Kaiqing Zhang, Georgios B. Giannakis, and Tamer Başar. Communication-Efficient Policy Gradient Methods for Distributed Reinforcement Learning. *IEEE Transactions on Control of Network Systems*, 9(2): 917–929, 2022. doi: 10.1109/TCNS.2021.3078100. 2.2.1

[12] Xiongjun Chen, Yiming Jiang, and Chenguang Yang. Stiffness Estimation and Intention Detection for Human-Robot Collaboration. In *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1802–1807, November 2020. doi: 10.1109/ICIEA48937.2020.9248186. ISSN: 2158-2297. 1

[13] Pravin Chopade, Saad M Khan, David Edwards, and Alina von Davier. Machine Learning for Efficient Assessment and Prediction of Human Performance in Collaborative Learning Environments. In *2018 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–6, October 2018. doi: 10.1109/THS.2018.8574203. 1

[14] Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 10707–10717. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/7967cc8e3ab559e68cc944c44b1cf3e8-Paper.pdf. 2.2.2

[15] Filippos Christianos, Georgios Papoudakis, Muhammad A. Rahman, and Stefano V. Albrecht. Scaling Multi-Agent Reinforcement Learning with Selective Parameter Sharing. In *Proceedings of the 38th International Conference on Machine Learning*, pages 1989–1998. PMLR, July 2021. URL https://proceedings.mlr.press/v139/christianos21a.html. ISSN: 2640-3498. 2.2.2

[16] Tianshu Chu, Sandeep Chinchali, and Sachin Katti. Multi-agent Reinforcement Learning for Networked System Control. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Syx7A3NFvH. 1

[17] Xiangxiang Chu and Hangjun Ye. Parameter sharing deep deterministic policy

gradient for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:1710.00336*, 2017. 2.2.2

[18] Caitlyn Clabaugh, Konstantinos Tsiakas, and Maja Mataric. Predicting preschool mathematics performance of children with a socially assistive robot tutor. In *Proceedings of the Synergies between Learning and Interaction Workshop@ IROS, Vancouver, BC, Canada*, pages 24–28, 2017. 1

[19] Christopher C. Corral, Keerthi Shrikar Tatapudi, Verica Buchanan, Lixiao Huang, and Nancy J. Cooke. Building a Synthetic Task Environment to Support Artificial Social Intelligence Research. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 65(1):660–664, September 2021. ISSN 2169-5067. doi: 10.1177/1071181321651354a. URL https://doi.org/10.1177/1071181321651354a. Publisher: SAGE Publications Inc. 1, 5.3.2, A

[20] Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. Adaptively Sparse Transformers. *CoRR*, abs/1909.00015, 2019. URL http://arxiv.org/abs/1909.00015. B.4

[21] Felipe Leno Da Silva and Anna Helena Reali Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703, 2019. 1

[22] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pages 1538–1546. PMLR, 2019. 1

[23] Jilles Dibangoye and Olivier Buffet. Learning to act in decentralized partially observable MDPs. In *International Conference on Machine Learning*, pages 1233–1242. PMLR, 2018. 2.2.1

[24] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy. B.4

[25] Simon S Du, Sham M Kakade, Ruosong Wang, and Lin F Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *International Conference on Learning Representations*, 2020. 1

[26] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran

Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/hash/c7635bfd99248a2cdef8249ef7bfbef4-Abstract.html. 1, 2.2.2

[27] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. Issue: 1. 2.2.1, 2.3.2

[28] Alison Freebairn, Kirsten Hagon, Vincent Turmine, Guido Pizzini, Roop Singh, Tessa Kelly, Catalina Jaime, Nikolas Scherer, Kara Siahaan, Julia Hartelius, et al. *World Disasters Report 2020: Come Heat Or High Water*. International Federation of Red Cross and Red Crescent Societies, 2020. 1

[29] Jared T. Freeman, Lixiao Huang, Matt Woods, and Stephen J. Cauffman. Evaluating artificial social intelligence in an urban search and rescue task environment, November 2021. URL https://keep.lib.asu.edu/items/162284. 1, A

[30] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34: 20132–20145, 2021. B.4

[31] Scott Fujimoto, David Meger, and Doina Precup. Off-Policy Deep Reinforcement Learning without Exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2052–2062. PMLR, May 2019. URL https://proceedings.mlr.press/v97/fujimoto19a.html. ISSN: 2640-3498. 1

[32] Angus Fung, Long Yu Wang, Kaicheng Zhang, Goldie Nejat, and Beno Benhabib. Using Deep Learning to Find Victims in Unknown Cluttered Urban Search and Rescue Environments. *Current Robotics Reports*, 1(3):105–115, September 2020. ISSN 2662-4087. doi: 10.1007/s43154-020-00011-8. URL https://doi.org/10.1007/s43154-020-00011-8. 1

[33] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning. In *Proceedings of the Conference on Robot Learning*, pages 1025–1037. PMLR, May 2020. URL https://proceedings.mlr.press/v100/gupta20a.html. ISSN: 2640-3498. B.4

[34] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In Gita Sukthankar and Juan A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems*, Lecture Notes in Computer Science, pages 66–83, Cham, 2017. Springer International Publishing. ISBN 978-3-319-71682-4. doi: 10.1007/978-3-319-71682-4_5. 1, 2.2.2

[35] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004. 2.2.1

[36] Caroline E. Harriott and Julie A. Adams. Modeling Human Performance for Human–Robot Systems. *Reviews of Human Factors and Ergonomics*, 9(1): 94–130, November 2013. ISSN 1557-234X. doi: 10.1177/1557234X13501471. URL https://doi.org/10.1177/1557234X13501471. Publisher: SAGE Publications. 1

[37] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-2649-2. URL https://www.nature.com/articles/s41586-020-2649-2. Number: 7825 Publisher: Nature Publishing Group. B.2

[38] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968. ISSN 2168-2887. doi: 10.1109/TSSC.1968.300136. Conference Name: IEEE Transactions on Systems Science and Cybernetics. 5.2

[39] Matthew Hausknecht and Peter Stone. Deep Recurrent Q-Learning for Partially Observable MDPs. In *2015 AAAI Fall Symposium Series*, September 2015. URL https://www.aaai.org/ocs/index.php/FSS/FSS15/paper/view/11673. 2.1

[40] Yujiao Hu, Yuan Yao, and Wee Sun Lee. A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs. *Knowledge-Based Systems*, 204:106244, September 2020. ISSN 0950-7051. doi: 10.1016/j.knosys.2020.106244. URL https://www.sciencedirect.com/science/article/pii/S0950705120304445. 2.4

[41] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation Learning: A Survey of Learning Methods. *ACM Computing Surveys*, 50(2):21:1–21:35, April 2017. ISSN 0360-0300. doi: 10.1145/3054912. URL http://doi.org/10.1145/3054912. 1

[42] Shariq Iqbal and Fei Sha. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2961–2970. PMLR, May 2019. URL https://proceedings.mlr.press/v97/iqbal19a.html. ISSN: 2640-3498. 2.2.2, 4.2

[43] Vidhi Jain, Rohit Jena, Huao Li, Tejus Gupta, Dana Hughes, Michael Lewis, and Katia P. Sycara. Predicting Human Strategies in Simulated Search and Rescue Task. *CoRR*, abs/2011.07656, 2020. URL https://arxiv.org/abs/2011.07656. 1

[44] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International conference on machine learning*, pages 3040–3049. PMLR, 2019. 1

[45] Jiechuan Jiang and Zongqing Lu. Learning Attentional Communication for Multi-Agent Cooperation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 7254–7264. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/6a8018b3a00b69c008601b8becae392b-Paper.pdf. 2.2.2

[46] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 1

[47] Soummya Kar, José M. F. Moura, and H. Vincent Poor. $\cal Q \cal D$-Learning: A Collaborative Distributed Strategy for Multi-Agent Reinforcement Learning Through $\rm Consensus + \rm Innovations$. *IEEE Transactions on Signal Processing*, 61(7):1848–1862, April 2013. ISSN 1941-0476. doi: 10.1109/TSP.2013.2241057. Conference Name: IEEE Transactions on Signal Processing. 2.2.1

[48] Meha Kaushik, Nirvan Singhania, Phaniteja S., and K. Madhava Krishna. Parameter Sharing Reinforcement Learning Architecture for Multi Agent Driving. In *Proceedings of the Advances in Robotics 2019*, AIR 2019, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 978-1-4503-6650-2. doi: 10.1145/3352593.3352625. URL https://doi.org/10.1145/3352593.3352625. event-place: Chennai, India. 2.2.2

[49] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning Combinatorial Optimization Algorithms over Graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d9896106ca98d3d05b8cbdf4fd8b13a1-Abstract.html. 2.4

[50] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*, 2015. URL http://arxiv.org/abs/1412.6980. B

[51] Vijay Konda and John Tsitsiklis. Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper/1999/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html. 2.3.2

[52] Antonio Laverghetta Jr, Animesh Nighojkar, Jamshidbek Mirzakhalov, and John Licato. Can Transformer Language Models Predict Psychometric Properties? In *Proceedings of* SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics*, pages 12–25, 2021. 1

[53] Hoang M. Le, Yisong Yue, Peter Carr, and Patrick Lucey. Coordinated Multi-Agent Imitation Learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1995–2003. PMLR, July 2017. URL https://proceedings.mlr.press/v70/le17a.html. ISSN: 2640-3498. 2.2.2

[54] Fuhao Li, Shike Hou, Chunguang Bu, and Bo Qu. Rescue Robots for the Urban Earthquake Environment. *Disaster Medicine and Public Health Preparedness*, pages 1–5, June 2022. ISSN 1935-7893, 1938-744X. doi: 10.1017/dmp.2022.98. URL http://www.cambridge.org/core/journals/disaster-medicine-and-public-health-preparedness/article/rescue-robots-for-the-urban-earthquake-environment/AE4E401B0D089C78EDB0DB635768D93A. Publisher: Cambridge University Press. 1

[55] Qingbiao Li, Weizhe Lin, Zhe Liu, and Amanda Prorok. Message-Aware Graph Attention Networks for Large-Scale Multi-Robot Path Planning. *IEEE Robotics and Automation Letters*, 6(3):5533–5540, July 2021. ISSN 2377-3766. doi: 10.1109/LRA.2021.3077863. Conference Name: IEEE Robotics and Automation Letters. 1, 2.3.2

[56] Amarildo Likmeta, Alberto Maria Metelli, Andrea Tirinzoni, Riccardo Giol, Marcello Restelli, and Danilo Romano. Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving. *Robotics and Autonomous Systems*, 131:103568, September 2020. ISSN 0921-8890. doi: 10.1016/j.robot.2020.103568. URL https://www.sciencedirect.com/science/article/pii/S0921889020304085. 1

[57] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier, 1994. 2.2.1

[58] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. Multi-agent game abstraction via graph attention neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7211–7218, 2020. Issue: 05. 1, 2.2.2, 2.3.2, 4.2, 4.3.1

[59] Yugang Liu and Goldie Nejat. Robotic Urban Search and Rescue: A Survey from the Control Perspective. *Journal of Intelligent & Robotic Systems*, 72(2):147–165, November 2013. ISSN 0921-0296, 1573-0409. doi: 10.1007/s10846-013-9822-x. URL http://link.springer.com/10.1007/s10846-013-9822-x. 1

[60] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html. 2.2.2, 2.3.1, 2.3.2

[61] Weifeng Lu, Zhe Hu, and Jia Pan. Human-Robot Collaboration using Variable Admittance Control and Human Intention Prediction. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1116–1121, August 2020. doi: 10.1109/CASE48305.2020.9217040. ISSN: 2161-8089. 1

[62] Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting Centralized and Decentralized Critics in Multi-Agent Reinforcement Learning, December 2021. URL http://arxiv.org/abs/2102.04402. arXiv:2102.04402 [cs]. 2.3.2

[63] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. MAVEN: Multi-Agent Variational Exploration. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/f816dc0acface7498e10496222e9db10-Abstract.html. 1, 2.2.2, 4.3.1

[64] Hangyu Mao, Zhengchao Zhang, Zhen Xiao, Zhibo Gong, and Yan Ni. Learning Agent Communication under Limited Bandwidth by Message Pruning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5142–5149, April 2020. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v34i04.5957. URL https://aaai.org/ojs/index.php/AAAI/article/view/5957. 1

[65] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, October 2021. ISSN 0305-0548. doi: 10.1016/j.cor.2021.105400. URL https://www.sciencedirect.com/science/article/pii/S0305054821001660. 2.4

[66] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. Publisher: Nature Publishing Group. 2.3.1

[67] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1928–1937. PMLR, June 2016. URL https://proceedings.mlr.press/v48/mniha16.html. ISSN: 1938-7228. 2.3.2, 4.2

[68] Kevin P Murphy. A survey of pomdp solution techniques. *environment*, 2:X3, 2000. 2.1

[69] Venkatraman Narayanan, Bala Murali Manoghar, Rama Prashanth RV, and Aniket Bera. EWareNet: Emotion Aware Human Intent Prediction and Adaptive Spatial Profile Fusion for Social Robot Navigation, December 2020. URL http://arxiv.org/abs/2011.09438. arXiv:2011.09438 [cs]. 1

[70] Yaru Niu, Rohan Paleja, and Matthew Gombolay. Multi-Agent Graph-Attention Communication and Teaming. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 964–973, 2021. 1, 2.3.2

[71] Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere. Game Theory and Multi-agent Reinforcement Learning. In Marco Wiering and Martijn van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, Adaptation, Learning, and Optimization, pages 441–470. Springer, Berlin, Heidelberg, 2012. ISBN 978-3-642-27645-3. doi: 10.1007/978-3-642-27645-3_14. URL https://doi.org/10.1007/978-3-642-27645-3_14. 2.2.2

[72] Paul D O'Brien and Richard C Nicol. FIPA—towards a standard for software agents. *BT Technology Journal*, 16(3):51–59, 1998. Publisher: Springer. 5.2

[73] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An Algorithmic Perspective on Imitation Learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, March 2018. ISSN 1935-8253, 1935-8261. doi: 10.1561/2300000053. URL http://www.nowpublishers.com/article/Details/ROB-053. Publisher: Now Publishers, Inc. 1

[74] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games, September 2017. URL http://arxiv.org/abs/1703.10069. arXiv:1703.10069 [cs]. 2.2.2

[75] Emanuele Pesce and Giovanni Montana. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. *Machine Learning*, 109(9):1727–1747, September 2020. ISSN 1573-0565. doi: 10.1007/s10994-019-05864-5. URL https://doi.org/10.1007/

s10994-019-05864-5. 2.2.2, 4.2

[76] Huy Xuan Pham, Hung Manh La, David Feil-Seifer, and Aria Nefian. Cooperative and Distributed Reinforcement Learning of Drones for Field Coverage, September 2018. URL http://arxiv.org/abs/1803.07250. arXiv:1803.07250 [cs]. 2.2.2

[77] François Pomerleau, Benoit Lescot, Francis Colas, Ming Liu, and Roland Siegwart. Dataset Acquisitions for USAR Environments. In *2011 AAAI Fall Symposium Series*, November 2011. URL http://www.aaai.org/ocs/index.php/FSS/FSS11/paper/view/4185. 1

[78] Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *arXiv preprint arXiv:2203.01387*, 2022. 1

[79] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. 2.1

[80] Peng Qian, Tahira Naseem, Roger Levy, and Ramón Fernandez Astudillo. Structural Guidance for Transformer Language Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3735–3745, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.289. URL https://aclanthology.org/2021.acl-long.289. 1

[81] Calvin Z Qiao, Maram Sakr, Katharina Muelling, and Henny Admoni. Learning from demonstration for real-time user goal prediction and shared assistive control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3270–3275. IEEE, 2021. 1

[82] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017. B.4

[83] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018. 1, 2.2.1, 2.2.2, 2.3.1, 4.3.1

[84] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 10199–10210. Curran Asso-

ciates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/73a427badebe0e32caa2e1fc7530b7f3-Abstract.html. 2.2.2

[85] Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging Offline Reinforcement Learning and Imitation Learning: A Tale of Pessimism. In *Advances in Neural Information Processing Systems*, volume 34, pages 11702–11716. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/60ce36723c17bbac504f2ef4c8a46995-Abstract.html. 1

[86] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. In *AAMAS*, pages 2186–2188, 2019. URL http://dl.acm.org/citation.cfm?id=3332052. 1, 2.2.2, 2.3.1, 3.1

[87] Stefano Scheggi and Gionata Salvietti. Haptic guidance in urban search and rescue scenarios with reduced visibility. In *Proc. 20th IMEKO TC4 Int. Symp. and 18th Int. Workshop on ADC Modelling and Testing Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy*, 2014. 1

[88] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1506.02438. 2.3.2, 4.2

[89] Francesco Semeraro, Alexander Griffiths, and Angelo Cangelosi. Human-robot collaboration and machine learning: a systematic review of recent research, May 2022. URL http://arxiv.org/abs/2110.07448. arXiv:2110.07448 [cs]. 1

[90] Esmaeil Seraj, Zheyuan Wang, Rohan Paleja, Matthew Sklar, Anirudh Patel, and Matthew Gombolay. Heterogeneous Graph Attention Networks for Learning Diverse Communication. Technical Report arXiv:2108.09568, arXiv, October 2021. URL http://arxiv.org/abs/2108.09568. arXiv:2108.09568 [cs] type: article. 1, 2.2.1, 2.2.2, 2.3.2

[91] Binoy Shah and Howie Choset. Survey on Urban Search and Rescue Robots. *Journal of the Robotics Society of Japan*, 22(5):582–586, July 2004. ISSN 0289-1824, 1884-7145. doi: 10.7210/jrsj.22.582. URL https://www.jstage.jst.go.jp/article/jrsj1983/22/5/22_5_582/_article/-char/ja/. Publisher: The Robotics Society of Japan. 1

[92] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953. 2.2.1

[93] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Individualized Controlled Continuous Communication Model for Multiagent Cooperative and Competitive Tasks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rye7knCqK7. 1, 2.3.2

[94] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5887–5896. PMLR, May 2019. URL https://proceedings.mlr.press/v97/son19a.html. ISSN: 2640-3498. 1, 2.2.2

[95] Ivan Sorokin, Alexey Seleznev, Mikhail Pavlov, Aleksandr Fedorov, and Anastasiia Ignateva. Deep Attention Recurrent Q-Network. Technical Report arXiv:1512.01693, arXiv, December 2015. URL http://arxiv.org/abs/1512.01693. arXiv:1512.01693 [cs] type: article. 2.1

[96] Matthijs T. J. Spaan. Partially Observable Markov Decision Processes. In Marco Wiering and Martijn van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, Adaptation, Learning, and Optimization, pages 387–414. Springer, Berlin, Heidelberg, 2012. ISBN 978-3-642-27645-3. doi: 10.1007/978-3-642-27645-3_12. URL https://doi.org/10.1007/978-3-642-27645-3_12. 2.1

[97] M. Statheropoulos, A. Agapiou, G. C. Pallis, K. Mikedi, S. Karma, J. Vamvakari, M. Dandoulaki, F. Andritsos, and C. L. Paul Thomas. Factors that affect rescue time in urban search and rescue (USAR) operations. *Natural Hazards*, 75(1): 57–69, January 2015. ISSN 1573-0840. doi: 10.1007/s11069-014-1304-3. URL https://doi.org/10.1007/s11069-014-1304-3. 1

[98] Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/hash/55b1927fdafef39c48e5b73b5d61ea60-Abstract.html. 1, 4.3.1

[99] Yanchao Sun, Ruijie Zheng, Parisa Hassanzadeh, Yongyuan Liang, Soheil Feizi, Sumitra Ganesh, and Furong Huang. Certifiably Robust Policy Learning against Adversarial Communication in Multi-agent Systems, July 2022. URL http://arxiv.org/abs/2206.10158. arXiv:2206.10158 [cs]. 2.2.2

[100] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-Decomposition Networks For

Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pages 2085–2087, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems. event-place: Stockholm, Sweden. 2.3.1

[101] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 1998. ISBN 978-0-262-19398-6. 2.1, 2.3.2

[102] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL https://proceedings.neurips.cc/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html. 2.3.2

[103] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993. 2.2.2, 2.3.1

[104] J. K. Terry, Nathaniel Grammel, Sanghyun Son, and Benjamin Black. Parameter Sharing For Heterogeneous Agents in Multi-Agent Reinforcement Learning. Technical Report arXiv:2005.13625, arXiv, January 2022. URL http://arxiv.org/abs/2005.13625. arXiv:2005.13625 [cs, stat] type: article. 2.2.2

[105] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, \Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 5.2, B.4

[106] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ. 1, 4.2, 4.2

[107] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9876–9886. PMLR, July 2020. URL https://proceedings.mlr.press/v119/wang20f.html. 2.2.2

[108] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge United Kingdom, 1989. 2.3.1

[109] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/

BF00992696. 2.3.2

[110] Jianqiong Xiao and Zhiyong Zhou. Research Progress of RNN Language Model. In *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pages 1285–1288, June 2020. doi: 10.1109/ICAICA50127.2020.9182390. 1

[111] Can Xu, Wanzhong Zhao, Jinqiang Liu, Chunyan Wang, and Chen Lv. An Integrated Decision-Making Framework for Highway Autonomous Driving Using Combined Learning and Rule-Based Algorithm. *IEEE Transactions on Vehicular Technology*, 71(4):3621–3632, April 2022. ISSN 1939-9359. doi: 10.1109/TVT.2022.3150343. Conference Name: IEEE Transactions on Vehicular Technology. 1

[112] Liang Yan, Xiaoshan Gao, Xiongjie Zhang, and Suokui Chang. Human-Robot Collaboration by Intention Recognition using Deep LSTM Neural Network. In *2019 IEEE 8th International Conference on Fluid Power and Mechatronics (FPM)*, pages 1390–1396, April 2019. doi: 10.1109/FPM45753.2019.9035907. 1

[113] Yang Ye and Shihao Ji. Sparse Graph Attention Networks. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021. ISSN 1558-2191. doi: 10.1109/TKDE.2021.3072345. Conference Name: IEEE Transactions on Knowledge and Data Engineering. B.4

[114] Javier Yu, Joseph A Vincent, and Mac Schwager. Dinno: Distributed neural network optimization for multi-robot collaborative learning. *IEEE Robotics and Automation Letters*, 7(2):1896–1903, 2022. 1

[115] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020. 1

[116] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. Heterogeneous Graph Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 793–803, New York, NY, USA, July 2019. Association for Computing Machinery. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330961. URL https://doi.org/10.1145/3292500.3330961. 2.4

[117] Gengzhi Zhang, Liang Feng, and Yaqing Hou. Multi-task Actor-Critic with Knowledge Transfer via a Shared Critic. In *Asian Conference on Machine Learning*, pages 580–593. PMLR, November 2021. URL https://proceedings.mlr.press/v157/zhang21b.html. ISSN: 2640-3498. 2.2.2, 2.3.2

[118] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents. In *Proceedings of the 35th International Conference on Machine Learning*, pages

5872–5881. PMLR, July 2018. URL https://proceedings.mlr.press/v80/zhang18n.html. ISSN: 2640-3498. 2.2.1, 2.2.2

[119] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. In Kyriakos G. Vamvoudakis, Yan Wan, Frank L. Lewis, and Derya Cansever, editors, *Handbook of Reinforcement Learning and Control*, Studies in Systems, Decision and Control, pages 321–384. Springer International Publishing, Cham, 2021. ISBN 978-3-030-60990-0. doi: 10.1007/978-3-030-60990-0_12. URL https://doi.org/10.1007/978-3-030-60990-0_12. 2.2.2

[120] Kuangen Zhang, Haiyuan Liu, Zixuan Fan, Xinxing Chen, Yuquan Leng, Clarence W. de Silva, and Chenglong Fu. Foot placement prediction for assistive walking by fusing sequential 3D gaze and environmental context. *IEEE Robotics and Automation Letters*, 6(2):2509–2516, 2021. Publisher: IEEE. 1

[121] Yu Zhang, Vignesh Narayanan, Tathagata Chakraborti, and Subbarao Kambhampati. A human factors analysis of proactive support in human-robot teaming. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3586–3593. IEEE, 2015. 1

[122] Liang Zheng, Chuang Zhu, Zhengbing He, and Tian He. Safety rule-based cellular automaton modeling and simulation under V2V environment. *Transportmetrica A: Transport Science*, 17(1):81–106, January 2021. ISSN 2324-9935. doi: 10.1080/23249935.2018.1517135. URL https://doi.org/10.1080/23249935.2018.1517135. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/23249935.2018.1517135. 1

[123] Lulu Zheng, Jiarui Chen, Jianhao Wang, Jiamin He, Yujing Hu, Yingfeng Chen, Changjie Fan, Yang Gao, and Chongjie Zhang. Episodic Multi-agent Reinforcement Learning with Curiosity-driven Exploration. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=YDGJ5YExiw6. 1

[124] Hao Zhou, Yazhou Yang, Tingjin Luo, Jun Zhang, and Shuohao Li. A unified deep sparse graph attention network for scene graph generation. *Pattern Recognition*, 123:108367, March 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2021.108367. URL https://www.sciencedirect.com/science/article/pii/S0031320321005471. B.4

[125] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020. 1

[126] Vittorio Amos Ziparo, Alexander Kleiner, Alessandro Farinelli, Luca Marchetti, and Daniele Nardi. Cooperative exploration for USAR robots with indirect

communication. *IFAC Proceedings Volumes*, 40(15):554–559, 2007. Publisher: Elsevier. 1

[127] Aaron Zweig, Nesreen Ahmed, Theodore L. Willke, and Guixiang Ma. Neural Algorithms for Graph Navigation. In *Learning Meets Combinatorial Algorithms at NeurIPS2020*, 2020. URL https://openreview.net/forum?id=sew79Me0W0c. 2.4