Coordination Machines for Minimizing Communication in Multi-Agent Reinforcement Learning

Jacob Heglund
jheglun2@illinois.edu
Department of Aerospace Engineering
University of Illinois Urbana-Champaign

Huy Tran
huytran1@illinois.edu
Department of Aerospace Engineering
University of Illinois Urbana-Champaign

Abstract

Multi-agent systems (MAS) are a promising solution for many real-world problems. However, complete and perfect communications are not guaranteed in the real-world. A key problem in MAS is thus ensuring that agents can meet global specifications while minimizing communication cost. We approach this problem by proposing a hierarchical model that decomposes a global, multi-agent task given by a linear temporal logic (LTL) formula into an equivalent automaton defined over subtasks given as LTL formulae. Our key idea is that each subtask may require different levels of inter-agent communication, and that modeling a global task as the composition of subtasks enables context-aware communication. To solve this problem, we formulate a hierarchical agent which solves an optimization problem to ensure an MAS satisfies a probabilistic task-completion specification while minimizing inter-agent communication. We then develop an algorithm to optimize the hierarchical agent based on the performance of a low-level team of agents that are trained to solve subtasks using multi-agent reinforcement learning (MARL). We then compare our approach to a baseline monolithic task model that approximates a common formulation used in MARL, and show our method provides a significant reduction in communication cost compared to the baseline.

1 Introduction

Autonomous agents typically act under conditions of incomplete observations, which makes communication an important mechanism to enable multiple agents to coordinate their current and future actions. However, key questions in the design of communication policies, such as when to communicate, what to communicate, and with which other agents to communicate, remain open research questions Zhu et al. (2024). Two broad classes of methods to learn multi-agent communication include those that define communication policies with a fixed global parameter Böhmer et al. (2020); Wang et al. (2022); Kim et al. (2019), and those that learn communication end-to-end as a function of state Kim et al. (2019); Li et al. (2021). These classes exist on opposite ends of a spectrum, where methods in the first class may struggle to model context-aware communication, while methods in the second class may struggle with the sample efficiency of learning an optimal communication policy. We propose a method somewhere in-between these two classes which uses a hierarchical model to decompose a global task into subtasks, then learns optimal communication within each subtask. This design choice ensures our method is context-aware, while trading off global optimality for sample efficiency and satisfactory performance.

Our goal is design a method to train a team of agents to successfully complete a task with probability of at least p_c while minimizing inter-agent communication. To accomplish this, we design a hierarchical model of a team of agents that we refer to as a coordination machine, which decomposes a global task into an equivalent, probabilistic automaton defined over subtasks. We then use the coordination machine to formulate a hierarchical agent that solves an optimization problem to ensure the MAS minimizes the inter-agent communication needed to satisfy a global probabilistic task-completion specification. Finally, we show that our approach provides a median 36% reduction in communication cost across the sampled values of p_c compared to a baseline that approximates a commonly-used formulation of tasks in MARL.

2 Related Work

There are several approaches to designing communication between agents in MAS. One class of methods involves learning an optimized message policy to control messages sent between agents based on a global parameter. SchedNet Kim et al. (2019) approaches this problem by optimizing a message scheduler and message policy based on a global bandwidth constraint. Meanwhile, Böhmer et al. (2020) approaches this problem by assuming a fixed coordination graph topology that defines inter-agent communication. Finally, Wang et al. (2022) learns a dynamic coordination graph topology, but defines a global upper-bound on communication. One issue with these methods is that they rely on at least one global parameter to define their communication policies (i.e., a fixed graph topology, a fixed upper-bound on communication bandwidth, etc.), and therefore cannot model communication policies where these parameters may vary based on context.

Another class of methods focuses on end-to-end learning of agent communication as a function of state. Singh et al. (2019) uses a gating mechanism to learn when to communicate between agents, and assumes environments with individual rewards to develop a method that can scale to many agents. While this method shows improved performance compared to competitive baselines, their use of individual rewards does not address environments where the reward function cannot be decomposed at the level of individual agents. Meanwhile, Niu et al. (2021) and Li et al. (2021) use variations of a graph attention network to learn contextual communication as a function of the joint state, and are not subject to the individual reward constraint. However, one key issue with these methods is that communication policies which are learned end-to-end present roadblocks in developing models of the contexts in which certain levels of communication are needed to achieve a given level of performance. This is not desirable in designing real-world communication networks, since, without an understanding of these contexts, resources would have to be allocated to provide the capability of maximum communication bandwidth at all times, which may waste resources that may be better-used elsewhere. Please see Zhu et al. (2024) and Yuan et al. (2023) for in-depth reviews of communication in MAS.

Early frameworks for studying hierarchical models in RL, such as semi-MDPs and options, typically formulate abstracted actions as sequences of low-level actions. However, these methods typically do not define global relationships between abstracted actions, and are therefore of little use for ensuring global properties of MAS Hutsebaut-Buysse et al. (2022). Meanwhile, work in communities such as formal methods, supervisory control, and concurrent computing has focused on developing frameworks to prove such properties Cassandras & Lafortune (2008), typically by defining an abstracted state space which tracks an MAS's behavior by creating an isomorphism between the abstracted state and truth-values of sentences about the MAS expressed in a system of formal logic. This approach can easily model non-Markovian tasks Icarte et al. (2018), can enable compositionality of tasks Jothimurugan et al. (2021), and can be used to prove global properties such as verifiable performance Neary et al. (2022) and safety Anderson et al. (2020). One weakness of these approaches is they have not been developed for use in designing multi-agent communication policies in complex environments.

3 Preliminaries

3.1 Multi-Agent Reinforcement Learning

One common approach to extending RL to a multi-agent setting is a Decentralized POMDP (Dec-POMDP) Oliehoek & Amato (2016). A Dec-POMDP is defined by a tuple $\mathcal{M} = (\mathcal{D}, \mathcal{S}, b_0, \mathcal{A}, P, \mathcal{O}, O, R, T, \gamma)$ where $\mathcal{D} = \{1, \ldots, n\}$ is the set of agents, $\mathcal{S}$ is the set of joint states,

$b_0 \in \Delta(\mathcal{S})$ is the joint state distribution at time 0, $\mathcal{A}$ is the joint action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition probability function, $\mathcal{O}$ is the set of observations, O is the set of observation probability functions, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the global reward function shared by all agents, T is the time horizon, and $\gamma \in (0, 1]$ is the discount factor. A common goal in multi-agent reinforcement learning is to learn an optimal joint policy, $\pi^* : \mathcal{S} \to \Delta(\mathcal{A})$, which maximizes the expected sum of discounted joint rewards over an infinite horizon, $G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$, where $r_t = R(s_t, a_t, s_{t+1})$.

3.2 Formal methods for task decomposition

Systems of formal logic provide useful foundations for reasoning about behaviors of autonomous systems. In particular, linear temporal logic (LTL) Pnueli (1977), is a modal temporal logic that extends propositional logic by introducing temporal modal operators such as *next* ($\circ$), *always* ($\Box$), *until* ($\mathcal{U}$), and *eventually* ($\diamond$) Wongpiromsarn et al. (2023). LTL formulae can also be translated into an equivalent automaton Gastin & Oddoux (2001), which provides a context-aware representation of a system's behavior that can be used to prove global properties such as verifiable performance Neary et al. (2022).

4 Coordination Machines

Our goal is to design a method to train and verify that a team of agents can successfully complete a task with a probability of at least p_c while minimizing inter-agent communication. We approach this problem by designing the hierarchical agent shown in Figure 1. This agent consists of four subsystems: **(A)** a high-level, probabilistic finite automaton (PFA) to model a global task's internal transition structure (i.e., transitions between subtasks), **(B)** a high-level agent to make decisions in the PFA, **(C)** a set of low-level, Dec-POMDP environments corresponding to subtasks in the PFA, and **(D)** a team of low-level agents trained to complete each subtask. This approach allows us to optimize the high-level agent based on the communication-dependent subtask success probabilities of the low-level agents. To motivate our approach, we introduce the environment shown in Figure 2 as a running example where the global task is for three agents to reach the goal states in the bottom right room. The agents begin in the top left room and progress through one of two possible paths, where each path requires that they cooperate to lift the red boxes which causes the blue doorways to open.

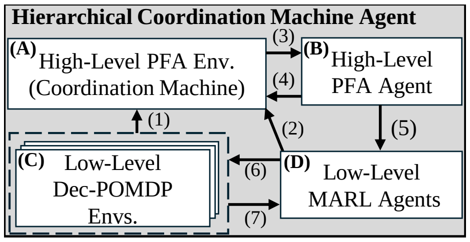
Hierarchical Coordination Machine Agent
(A) High-Level PFA Env. (Coordination Machine)
(B) High-Level PFA Agent
(C) Low-Level Dec-POMDP Envs.
(D) Low-Level MARL Agents
(1)
(2)
(3)
(4)
(5)
(6)
(7)

Figure 1: Interfaces between subsystems of the hierarchical coordination machine agent: (1) low-level environment model, (2) low-level policies, levels of communication, and success probabilities, (3) high-level environment model, (4) high-level actions and low-level policy indices, (5) low-level policy indices, (6) selected low-level actions, (7) rewards and observations.

4.1 Coordination Machines for Context-Aware Multi-Agent Communication

We begin by defining the high-level PFA environment model shown in Figure 1 **(A)**. PFAs provide a useful approach to model tasks as a composition of subtasks when there is uncertainty about a low-level team's capability to successfully complete a given subtask. Formally, a PFA is defined by a tuple $\mathcal{T} = (U, \Sigma, u_I, u_F, \delta)$ where U is a finite, nonempty set of PFA states, Σ is a finite set of input symbols, $u_I \in U$ is the initial state, $u_F \in U$ is the final state, and $\delta : U \times 2^{\Sigma} \to \Delta(U)$ is the partial probabilistic transition function Paz (1971). We assume the input string to the transition function consists of two symbols, $u'w$, where $u' \in U' := U \setminus u_X$ specifies the desired next state, u_X is a failure state, $w \in \{0, 1, \ldots, N_w\}$ is associated with a low-level policy to complete the transition (u, u'), and N_w is the number of communication levels considered for completing a given subtask. Due to our definition of U', the PFA will not accept a string that is intended to cause a failure; however, due to the probabilistic transition function, it is possible for the PFA to transition to u_X.

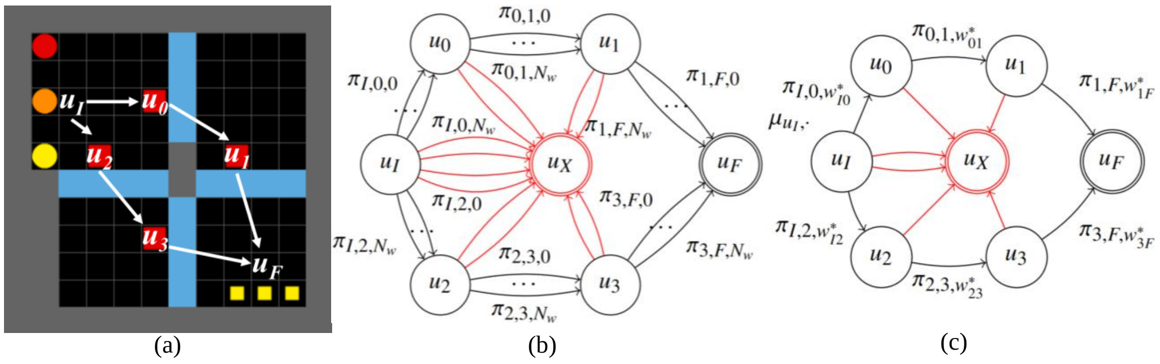

Figure 2: (a) Agents must reach the goal states in the bottom right room starting in the top left room. (b) The CM representation of the task prior to selection of μ and ν, where each (u, u') represents a subtask. (c) μ defines the high-level agent's policy in the CM, while ν selects a single low-level policy for each subtask.

We extend the definition of a PFA to include relevant information about low-level decision-making processes by specifying information about the low-level environments (Figure 1 **(C)**) and the set of low-level agents (Figure 1 **(D)**). Figure 1 (1) represents the two pieces of information that our extended PFA model needs about the low-level environments, namely, $\mathcal{M}_{uu'}$, the low-level environment model for transition (u, u') (i.e., a Dec-POMDP), and $\phi_{u'}$, the binary subtask-completion specification that must be satisfied for the PFA to transition into u'. Figure 1 (2) then represents information about a set of Markovian low-level agents that directly interact with $\mathcal{M}_{uu'}$ using actions selected by a joint policy, $\pi_{uu'w} \in \Pi_{uu'} := \{\pi_{uu'0}, \dots, \pi_{uu'N_w}\}$, where $\Pi_{uu'}$ is a set of N_w policies that can be selected from to complete transition (u, u') by satisfying $\phi_{u'}$. In our case, each w is associated with a low-level policy that satisfies $\phi_{u'}$ with probability $p_{uu'w}$ using a level of communication $\lambda_{uu'w}$. To organize the information about low-level decision-making processes, we assume the valid transitions in the PFA are given as the set of edges $\mathcal{E}$, where each edge, $e = (u, u', w) \in \mathcal{E}$ has properties $(u'w, \mathcal{M}_{uu'}, \phi_{u'}, \pi_{uu'w}, p_{uu'w}, \lambda_{uu'w})$, where $u'w$ is the entry condition for the edge. Given the novel extension of a PFA we have defined, we now define a coordination machine as follows:

Definition (Coordination Machine) A coordination machine (CM) is a tuple $\mathcal{C} = (\mathcal{T}, \mathcal{E})$ where $\mathcal{T}$ is a PFA and $\mathcal{E}$ is a set of edges that model low-level, coordinated, group decision-making processes in a low-level environment, where each edge has properties $(u'w, \mathcal{M}_{uu'}, \phi_{u'}, \pi_{uu'w}, p_{uu'w}, \lambda_{uu'w})$.

To model high-level decision-making in the CM, we assume the existence of a high-level agent (Figure 1 **(B)**) that has access to the CM environment model (Figure 1 (3)), and uses this model to output a string $u'w$ (Figure 1 (4) and (5)) in each state u such that the agent specifies the desired next state u' and the index w which selects a single low-level policy from $\Pi_{uu'}$ to complete (u, u') by satisfying $\phi_{u'}$. Due to the two-stage structure of decision-making, we define two decision-making functions, namely $\mu : U \to \Delta(U')$, the agent's stochastic action policy function, and $\nu : U \times U' \times \{0, \dots, N_w\} \to \{0, 1\}$, the agent's low-level policy-selection defined as an indicator function such that $\sum_{i=0}^{N_w} \nu(u, u', i) = 1$. Figure 2 then shows the CM before (b) and after (c) the high-level agent uses ν to select a single low-level policy for each transition. We now define the CM's transition function in terms of μ, ν, and $u'w$ as,

$$\delta(u, u', w) = \begin{cases} u' & \text{w.p.} \quad \mu(u' \mid u)\nu(u, u', w)p_{uu'w} \\ u_X & \text{w.p.} \quad \mu(u' \mid u)\nu(u, u', w)(1 - p_{uu'w}), \end{cases} \tag{1}$$

where we assume μ and ν are obtained prior to the high-level agent's initialization in u_I, and that they are stationary, therefore inducing a Markov chain.

We now define the high-level agent as a goal-directed agent by defining the CM as modeling the agent completing a task with probability of at least p_c. We start by defining the global task as an LTL reachability specification, $\diamond\phi_G$ (i.e., eventually ϕ_G is satisfied). We then define $\phi_{u_F} \coloneqq \phi_G$ such that a high-level agent navigating a sequence of CM states $(u_I, \ldots, u_F)$ is equivalent to satisfying ϕ_G. This implies that the probabilistic reachability specification $P(\diamond\phi_G) \geq p_c$ is equivalent to $P(\diamond(u = u_F) \mid \mathcal{C}, \mu, \nu) \geq p_c$. In our running example, we define subtasks with specifications $\phi_{u'} \in \{\text{lift\_box}_{u'}, \text{reach\_goal}\}$, and, as shown in Figure 2, we model two possible sequences that simulate the global task, namely, (u_I, u_0, u_1, u_F) and (u_I, u_2, u_3, u_F). Given the novel structure of the system we have defined, we now formally define the hierarchical coordination machine agent as follows:

Definition (Hierarchical Coordination Machine Agent) A hierarchical coordination machine agent (Figure 1) is a tuple $\mathcal{A}_\mathcal{C} = (\mathcal{C}, \mu, \nu, p_c)$ which satisfies $P(\diamond(u = u_F) \mid \mathcal{C}, \mu, \nu) \geq p_c$ while minimizing communication used by a low-level team of agents, where $\mathcal{C}$ is a CM, μ is the high-level agent's policy, and ν is the high-level agent's policy-selection function.

4.2 Optimizing the High-Level Agent's Policy and Policy-Selection Function

Our goal in this section is to formulate an optimization problem which ensures the hierarchical CM agent satisfies $P(\diamond(u = u_F) \mid \mathcal{C}, \mu, \nu) \geq p_c$ while minimizing global communication used by a team of low-level agents. We accomplish this by formulating a mixed-integer optimization problem which selects μ and ν to satisfy the task-completion specification, p_c. To model communication cost, we define $\lambda_{uu'w} \in [0, 1]$, where $\lambda_{uu'w}$ models both the amount of communication used by a policy and the cost of communication. Due to the modular nature of our hierarchical CM agent, many MARL methods that make use of inter-agent communication could be used as part of our method to train a team of low-level agents (Figure 1 **(D)**). In this work, we demonstrate our approach using CASEC Wang et al. (2022), where we formulate λ to model the number of directed edges in the coordination graph constructed by the algorithm. For other MARL algorithms, such as SchedNet Kim et al. (2019), λ could model some combination of the K_{sched} and L_{band} parameters that control the number of agents that are allowed to broadcast and the size of their messages.

$$\min_{x_{uu'}, \nu_{uu'w}} J = \tag{2}$$

$$\sum_{u \in U \setminus \{u_X, u_F\}} \sum_{u' \in U'} x_{uu'} \sum_{i=0}^{N_w} \nu_{uu'i} \lambda_{uu'i} \tag{3}$$

$$\min_{x_{uu'}, \nu_{uu'w}} \quad J = \sum_{u \in U \setminus \{u_X, u_F\}} \sum_{u' \in U'} x_{uu'} \sum_{i=0}^{N_w} \nu_{uu'i} \lambda_{uu'i} \tag{4}$$

$$\text{s.t.} \quad \sum_{i=0}^{N_w} \nu_{uu'i} = 1 \ \forall\ (u, u', w) \in \mathcal{E} \tag{5}$$

$$x_{uu'} \geq 0 \ \forall\ u \in U \setminus \{u_X, u_F\},\ \forall\ u' \in U' \tag{6}$$

$$\sum_{u' \in U'} x_{uu'} = [u = u_I] + \sum_{(u_p, u) \in \text{pred}(u)} x_{u_p u} \sum_{i=0}^{N_w} \nu_{u_p u i} p_{u_p u i} \ \forall\ u \in U \setminus \{u_X, u_F\} \tag{7}$$

$$\sum_{(u_p, u_F) \in \text{pred}(u_F)} x_{u_p u_F} \sum_{i=0}^{N_w} \nu_{u_p u_F i} p_{u_p u_F i} \geq p_c \tag{8}$$

Here, Equation (4) defines μ and ν as functions that minimize communication cost and Equation (5) defines $\nu_{uu'w} \in \{0, 1\}$ as an indicator function that selects a single low-level policy $\pi_{uu'w}$ from a set of N_w policies. Equation (6) and Equation (7) (i.e., the Bellman-flow constraint) define $x_{uu'}$ as selecting the expected number of times transition (u, u') is completed by a high-level agent that starts in u_I and uses policy μ, where $\mu(u, u') \coloneqq \frac{x_{uu'}}{\sum_{i \in U'} x_{ui}}$ and $\text{pred}(u)$ is the set of predecessor state-action pairs associated with state u. Finally, Equation (8) enforces the hierarchical CM agent's satisfaction of $P(\diamond(u = u_F) \mid \mathcal{C}, \mu, \nu) \geq p_c$. Please see Puterman (2014) and Etessami et al. (2008) for more details about this formulation.

4.3 Optimizing the Set of Low-Level Policies in a Hierarchical CM Agent

Our goal in this section is to define a training algorithm for the case of a partially-specified CM model. Here we assume the set of valid transitions in the CM is specified prior to training, but the transition probabilities $p_{uu'w}$, are unknown and must be estimated to fully-specify the CM model (Figure 1 (2)) prior to optimization of the high-level agent. We assume each low-level environment model $\mathcal{M}_{uu'}$ is a Dec-POMDP with a final joint state distribution, $b_{f,u'}$. We also assume the set of initial state distributions in the CM, $\mathcal{B}_0$, is fully specified prior to training as $\mathcal{B}_0 \coloneqq \{b_{0,u}\}_{u \in U}$. In our running example, we define each $b_{f,u'}$ over a single joint state $\boldsymbol{s}_{f,u'}$, which allows us to formally define the LTL specifications for our subtasks as $\text{lift\_box}_{u'} \coloneqq \diamond\Big((\boldsymbol{s} = \boldsymbol{s}_{f,u'}) \wedge (\text{box\_lifted}_{u'})\Big)$, and $\text{reach\_goal} \coloneqq \diamond(\boldsymbol{s} = \boldsymbol{s}_{f,u'})$.

Algorithm 1 Optimizing a Hierarchical CM Agent based on a Partially-Specified CM

```
1: **Input**: $\mathcal{C} = (\mathcal{T}, \mathcal{E}), \{b_{0,u}\}_{u \in U}, p_c, N_{\text{tr}}, N_{\text{tr, max}}, N_p$
2: **Output**: $\mu, \nu, \Pi_{t_{\max}}$
3: $\Pi, \mathcal{B}_0 = \text{InitPolicies}(\mathcal{C}), \{b_{0,u}\}_{u \in U}$
4: $\Pi_{t_{\max}}, \mathcal{P}_{t_{\max}} = \Pi, [0]_{e \in \mathcal{E}}$
5: for $e = (u, u', w) \in \mathcal{E}$ do in parallel
6:     while $t_e < N_{\text{tr, max}}$ do
7:         $\pi_e, t_e \leftarrow \text{TrainPolicy}(\pi_e, N_{\text{tr}}, \mathcal{B}_{0,u}), t_e + N_{\text{tr}}$
8:         $\hat{p}_e = \text{SampleEps}(\pi_e, N_p)$
9:         if $\hat{p}_e > \mathcal{P}_{t_{max},e}$ then
10:            $\Pi_{t_{max},e}, \mathcal{P}_{t_{max},e} \leftarrow \pi_e, \hat{p}_e$
11: $\mu, \nu = \text{HLOpt}(\mathcal{C}, \mathcal{P}_{t_{\max}}, p_c)$          ▷ Equations (4) to (8)
12: return $\mu, \nu, \Pi_{t_{\max}}$
```

We describe our algorithm to train low-level policies in Algorithm 1. Each iteration first trains a set of low-level policies for N_{tr} steps, then estimates the probability that each policy can complete a task successfully by sampling evaluation episodes. Here, we model the success or failure of an episode as a Bernoulli random variable with parameter p estimated as the maximum likelihood estimator, $\hat{p}_{uu'w} = 1/N_p \sum_{i=1}^{N_p} \Big[\phi_{u',i}\Big]\Big|_{\pi=\pi_{uu'w,t_{\max}}}$, where $[\cdot]$ denotes the Iverson bracket defined as $[P] = 1$ if P is True for declarative sentence P and 0 otherwise. We define $N_p \coloneqq \frac{z^2(1/2)(1-1/2)}{d^2}$ based on a worst-case number of sampled evaluation episodes (Thompson, 2012, Equation 5.2) to ensure $|\hat{p}_{uu'w} - p_{uu'w}| < d$ with confidence α where z is the upper $\alpha/2$ point of the standard normal distribution. During training, we identify the best-performing policies for each subtask, $\pi_{uu'w,t_{\max}}$, where $t_{\max} \coloneqq \arg\max_t \hat{p}_{uu'wt}$, which we then use to form the set of estimated success probabilities, $\mathcal{P}_{t_{\max}}$ and solve for μ and ν.

5 Experiments and Discussion

We evaluate our method by training a multi-agent team to solve the gridworld environment shown in Figure 2, where each agent can choose to move in the four cardinal directions, stay still, and attempt

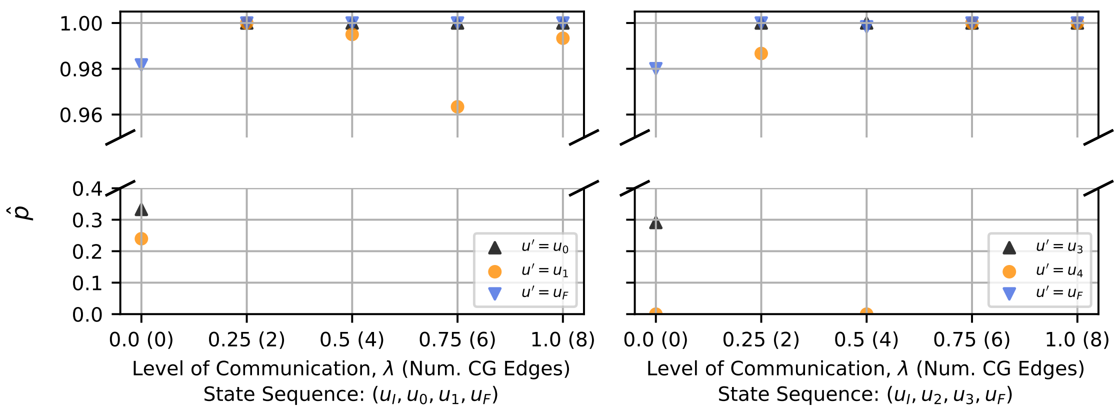

Figure 3: The agents learn near-optimal policies for most levels of communication, but do not learn optimal policies when no communication is used in the box-lifting subtasks that terminate in u_0, u_1, u_3 and u_4.

to lift an adjacent object. We model environment dynamics based on the Frozen Lake environment Towers et al. (2023) such that an agent will move in the selected direction with probability 95% and in a perpendicular direction with probability 2.5%, where agent movement order is randomly selected at each time step. We use the box-lifting dynamics shown in Table 1 to model subtasks that require different types of coordination, where $p_{\text{lift},N}$ is the probability the box will be successfully lifted if N agents attempt to simultaneously lift the box. We assign different boxes to the two modeled sequences of states to generate sequences of subtasks that may require different levels of communication to achieve a given success rate.

(u, u')	Box Type	$(p_{\text{lift},1}, p_{\text{lift},2}, p_{\text{lift},3})$
(u_I, u_0)	Large and heavy	(0.5, 0.75, 1)
(u_0, u_1)	Small and light	(0.7, 0.95, 0.7)
(u_I, u_2)	Large and light	(0.8, 0.9, 1)
(u_2, u_3)	Small and heavy	(0.5, 0.95, 0.5)

Table 1: Our environment's probabilistic box-lifting dynamics.

We train each policy using CASEC Wang et al. (2022), using the training algorithm outlined in Section 4.3, where λ corresponds to the five possible numbers of directed edges in the constructed coordination graph as shown in Figure 3. Since λ also represents the cost of communication, we choose values that are equally-spaced to ensure the marginal cost of communication is constant. We then train each policy for a total of $N_{\text{tr, max}} = 750,000$ steps, where each CM training iteration involves training the policies for $N_{\text{tr}} = 10,000$ steps, then evaluating for $N_p = 601$ episodes to estimate the success probability with a confidence level of $\alpha = 0.05$ and $d = 0.04$. We then train policies for two random seeds with all other hyperparameters held constant. Finally, we implement and solve the optimization problem described in Equations (4) to (8) using Gurobi 11.0 Gurobi Optimization, LLC (2024).

Figure 3 then shows the resulting estimated success probability for each level of communication and subtask. Most of the policies and levels of communication obtain near-optimal solutions, although a few runs fail to solve the task. This figure is useful to identify subtasks and levels of communication where hyperparameter tuning may be most-helpful to improve the system's overall performance.

To demonstrate the value of our hierarchical approach, which relies on decomposing tasks into subtasks to minimize communication (Figure 4 (a)), we compare our approach to a baseline in which a single level of communication is used across all subtasks. This baseline approximates a common formulation used in MARL works in which tasks are not decomposed into subtasks (Figure 4 (b)), and therefore the resulting policies are constrained to use a single level of communication for the entire task. To optimize the high-level agent using our method and the baseline method, we use the estimated success probabilities from Figure 3 to solve Equations (4) to (8), where the baseline is subject to an additional constraint such that $\nu_w = \nu_{uu'w} = \forall\ (u, u', \cdot) \in \mathcal{E}$. Figure 5 shows the results of this optimization across a range of p_c, where we note that the cost is guaranteed to monotonically increase as p_c increases.

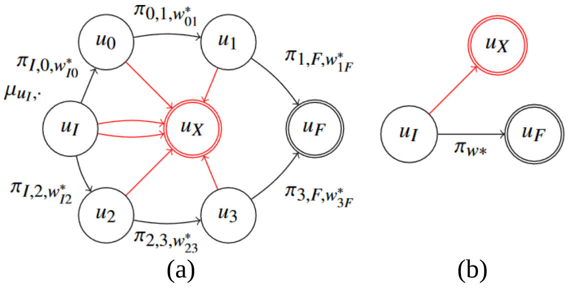

Figure 4: (a) We model context-aware communication by decomposing a global task into subtasks. (b) Our baseline approximates a monolithic task model.

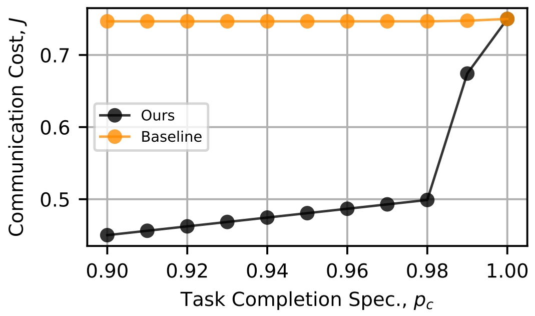

Figure 5: Our method incurs a cost less than or equal to the baseline across all sampled values of p_c.

We find our method incurs a cost that is less than or equal to the baseline at all sampled values of p_c, with a 36% median reduction in cost across the sampled values of p_c and a maximum reduction in cost of 40% for $p_c = 0.9$. We observe that the baseline selects $\lambda = 0.25$ for all subtasks and all levels of p_c, while our approach selects $\lambda_{u_I,u_0} = 0$ for $p_c \leq 0.98$, and $\lambda_{u_I,u_0} = 0.25$ for $p_c > 0.98$. This is because the baseline model has fewer degrees of freedom, and therefore cannot optimize communication based on the performance within each subtask. This result supports the key thesis of this work, namely, that decomposing a global task into subtasks allows us to ensure the MAS satisfies a given level of performance while minimizing inter-agent communication, while at the same time ensuring the global communication cost is less than or equal to a baseline monolithic model. We also find the increase in cost for our method is nonlinear. We note the communication cost increases by 35% when p_c increases from 0.98 to 0.99, which is significant compared to the increase in cost of 1.2% when p_c increases from 0.97 to 0.98. This is useful because it can be used to identify regions where small changes in p_c can cause large changes in the required communication. This behavior can be explained as a result of the formulation of Equations (4) to (8). In particular, because $\nu_{uu'w}$ is discrete and $x_{uu'}$ is continuous, the optimizer can select values of $\nu_{uu'w}$ that are optimal for a range of p_c, within which $x_{uu'}$ can vary to meet the probabilistic task-completion specification.

6 Conclusion

This work develops a novel hierarchical agent that satisfies a high-level probabilistic task-completion specification while minimizing communication between low-level agents. Using this model, we formulate an optimization problem that solves for the high-level agent's policy and policy selection function, and a training algorithm to train low-level MARL policies with varying communication cost. We demonstrate that our approach provides a significant reduction in communication cost compared to a baseline that approximates an approach used by many works in MARL. Future extensions of this work include more-extensive empirical validation of our method and developing a CM training algorithm to allow less-restrictive formulations of subtask specifications.

References

Greg Anderson, Abhinav Verma, Isil Dillig, and Swarat Chaudhuri. Neurosymbolic reinforcement learning with formally verified exploration. *Advances in neural information processing systems*, 33:6172–6183, 2020.

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pp. 980–991. PMLR, 2020.

Christos G Cassandras and Stéphane Lafortune. *Introduction to discrete event systems*. Springer, 2008.

Kousha Etessami, Marta Kwiatkowska, Moshe Y Vardi, and Mihalis Yannakakis. Multi-objective model checking of markov decision processes. *Logical Methods in Computer Science*, 4, 2008.

Paul Gastin and Denis Oddoux. Fast ltl to büchi automata translation. In *Computer Aided Verification: 13th International Conference, CAV 2001 Paris, France, July 18–22, 2001 Proceedings 13*, pp. 53–65. Springer, 2001.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL https://www.gurobi.com.

Matthias Hutsebaut-Buysse, Kevin Mets, and Steven Latré. Hierarchical reinforcement learning: A survey and open research challenges. *Machine Learning and Knowledge Extraction*, 4(1):172–221, 2022.

Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pp. 2107–2116. PMLR, 2018.

Kishor Jothimurugan, Suguman Bansal, Osbert Bastani, and Rajeev Alur. Compositional reinforcement learning from logical specifications. *Advances in Neural Information Processing Systems*, 34, 2021.

Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJxu5iR9KQ.

Sheng Li, Jayesh K Gupta, Peter Morales, Ross Allen, and Mykel J Kochenderfer. Deep implicit coordination graphs for multi-agent reinforcement learning. In *AAMAS*, pp. 764–772, 2021.

Cyrus Neary, Christos Verginis, Murat Cubuktepe, and Ufuk Topcu. Verifiable and compositional reinforcement learning systems. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pp. 615–623, 2022.

Yaru Niu, Rohan Paleja, and Matthew Gombolay. Magic: Multi-agent graph-attention communication. In *Mair2 Workshop at International Conference on Computer Vision (ICCV)*, 2021.

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

Azaria Paz. *Introduction to probabilistic automata*. Academic Press, 1971.

Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pp. 46–57. ieee, 1977.

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

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rye7knCqK7.

Steven K Thompson. *Sampling*, volume 755. John Wiley & Sons, 2012.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

Tonghan Wang, Liang Zeng, Weijun Dong, Qianlan Yang, Yang Yu, and Chongjie Zhang. Context-aware sparse deep coordination graphs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=wQfgfb8VKTn.

Tichakorn Wongpiromsarn, Mahsa Ghasemi, Murat Cubuktepe, Georgios Bakirtzis, Steven Carr, Mustafa O Karabag, Cyrus Neary, Parham Gohari, and Ufuk Topcu. Formal methods for autonomous systems. *arXiv preprint arXiv:2311.01258*, 2023.

Lei Yuan, Ziqian Zhang, Lihe Li, Cong Guan, and Yang Yu. A survey of progress on cooperative multi-agent reinforcement learning in open environment. *arXiv preprint arXiv:2312.01058*, 2023.

Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Autonomous Agents and Multi-Agent Systems*, 38(1):4, 2024.