



Title	On reward distribution in reinforcement learning of multi-agent surveillance systems with temporal logic specifications
Author(s)	Terashima, Keita; Kobayashi, Koichi; Yamashita, Yuh
Citation	Advanced Robotics, 38(6), 386-397 https://doi.org/10.1080/01691864.2024.2309634
Issue Date	2024-03-18
Doc URL	https://hdl.handle.net/2115/94010
Rights	This is an Accepted Manuscript of an article published by Taylor & Francis in Advanced Robotics on 05 Apr 2024, available at: https://doi.org/10.1080/01691864.2024.2309634 .
Type	journal article
File Information	advanced_robotics_terashima.pdf



On Reward Distribution in Reinforcement Learning of Multi-Agent Surveillance Systems with Temporal Logic Specifications

Keita Terashima^a, Koichi Kobayashi^a, and Yuh Yamashita^a

^aGraduate School of Information Science and Technology, Hokkaido University, Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan

ARTICLE HISTORY

Compiled January 19, 2024

ABSTRACT

In multi-agent systems, it is important to design a reward based on the contribution of each agent for efficient learning. In this paper, we propose a reward distribution method for a surveillance system based on our previously proposed multi-agent reinforcement learning method with an aggregator, in which a control specification is described by a linear temporal logic formula. In this method, the aggregator computes and distributes rewards according to the actions that agents take on the surveillance system. Finally, the effectiveness of the proposed method is presented through a numerical simulation of a surveillance problem addressing a specific type of linear temporal logic specification.

KEYWORDS

multi-agent reinforcement learning, linear temporal logic, surveillance systems, aggregator, reward distribution

1. Introduction

A multi-agent system (MAS) consists of a group of autonomous, interacting entities that share a common environment [1]. Within the area of cooperative MAS, it is a challenge how agents collaborate to achieve common goals. Reinforcement learning (RL), a robust learning method for unknown and uncertain environments, is one of the well-known learning methods used in MAS design [2]. RL is a highly effective method to design large, complex systems, as it does not require a designer to have any prior knowledge of the environment. In cooperative multi-agent reinforcement learning (MARL), the challenge is how to distribute rewards among agents for achieving a shared task. Inappropriate reward design makes it difficult for agents to learn cooperative behaviors and achieve a desirable control policy.

On the other hand, one recent trend is to use linear temporal logic (LTL) [3] to describe control specifications, and to design controllers through RL under temporal logic constraints. Temporal logic, an extended logic system that incorporates temporal operators into standard logic operations, enables the clarification of time-related properties. Various MARL methods have been studied using several types of temporal logic [4–7]. In addition, when applying temporal logic to MAS, some studies assign local LTL specifications to individual agents aside from global LTL specifications, as in [6–9]. Whereas, others focus on the problem setting where a cooperative group of agents achieves a single, global LTL specification, as in [4,5,10,11].

In our previous work [12], we have proposed a novel MARL method with an aggregator, where the control specification is described by an LTL formula. In this work, we focused on extending [13] developed for the single-agent framework to MASs. It is well known that technical issues generally arise in MARL, in particular the state space explosion problem, causing a significant reduction in learning efficiency. In addressing this problem, as detailed in [12], we propose a solution through the incorporation of an aggregator, which enables the partitioning of an environment and an evaluation function for each agent. In cooperative MARL algorithms, there are several known methods where a kind of central learners or aggregators is introduced during training, e.g., centralized training with decentralized execution (CTDE) [14], value-decomposition networks (VDN) [15], counterfactual multi-agent (COMA) [16], or duPLEX dueling multi-agent Q-learning (QPLEX) [17], etc. However, to the best of our knowledge, the existing methods have not taken temporal logic into account.

One of the issues in [12] is the low efficiency of acquiring rewards relative to the number of agents, and it is necessary to improve the learning performance by devising a way to give rewards to a group of agents. In reward design in cooperative MARL, various studies have addressed reward distribution methods in areas such as path planning or motion planning, as in [18–21]. However, reward design for MARL with LTL specifications and aggregators has not been studied.

In this paper, we propose a reward distribution method for a surveillance system with our previously proposed MARL method. The reward distribution method refers to a mechanism that adjusts the amount of rewards to individual agents according to their respective contribution levels. Higher contributions receive a greater reward, whereas lower contributions receive comparatively a lesser reward. First, after some preliminaries on an environment and control specifications given to agents, we introduce our MARL method with an aggregator. Next, we propose a reward distribution method for a surveillance problem where a group of mobile robots patrols multiple checkpoints. When at least one robot completes a task, their rewards are calculated and distributed according to the distance from the checkpoints. Finally, we compare numerical simulation results for some reward design cases, where the control specification is described by a specific type of LTL formula, represented by the conjunction of **GF** (Globally Future) and **G** (Globally).

The rest of this paper is structured as follows. A Markov decision process (MDP), LTL, and the conversion from LTL to automata are briefly explained in Section 2. In Section 3, we introduce our previous MARL method using an aggregator. In Section 4, we propose a reward distribution method for a surveillance problem and present some cases of reward design. In Section 5, we present a simulation example that illustrates the performance of the proposed method. Finally, conclusions are presented in Section 6.

Notation: $\mathbb{N}_0$ is the set of nonnegative integers. $\mathbb{R}_{\geq 0}$ is the set of nonnegative real numbers. ε is the empty string.

2. Preliminaries

In this section, we explain the necessary preparations for the introduction of our MARL method in Section 3. First, we present an overview of the MDP concerning the case of only one agent. Second, we delineate LTL, which expresses control specifications given to the agents. Then, we describe a limit-deterministic generalized Büchi automaton (LDGBA), which is obtained by converting from an LTL formula.

2.1. Markov Decision Process

In this paper, we assume that a given surveillance system can be modeled as an MDP defined below.

Definition 2.1. In the case of a single agent, a (labeled) MDP is a tuple $M = (S, A, s_{init}, P, AP, L)$, where S is a finite set of states, A is a finite set of actions, $s_{init} \in S$ is the initial state, $P : S \times A \times S \rightarrow [0, 1]$ is a transition probability function, AP is a finite set of atomic propositions, and $L : S \times A \times S \rightarrow 2^{AP}$ is a labeling function that assigns a subset of atomic propositions to each state transition. Let $\mathcal{A}(s) = \{a \in A; \exists s' \in S \text{ s.t. } P(s'|s, a) \neq 0\}$ denote an action space selectable in a state $s \in S$. For any state $s \in S$ and action $a \in \mathcal{A}(s)$, $\sum_{s' \in S} P(s'|s, a) = 1$ holds.

We present an example of an MDP for a surveillance problem.

Example 2.2. Consider the 7×7 grid world shown in Fig. 1 as the environment of the surveillance system given to a mobile robot. Each element of MDP M can be expressed as follows:

- $S = \{R_i\}_{i=1}^{49}$.
- $A = \{\text{Up, Down, Right, Left}\}$.
- $s_{init} = R_{43}$.
- $P(s'|s, a)$: The agent moves in the intended direction with a probability of $p \in [0, 1]$, while it stays in the same state with a probability of $1 - p$.
- $AP = \{r_4, r_{23}, r_{35}, r_{38}, u\}$.
- $L((s, a, s'))$ is given by

$$L((s, a, s')) = \begin{cases} \{r_4\} & \text{if } s' = R_4, \\ \{r_{23}\} & \text{if } s' = R_{23}, \\ \{r_{35}\} & \text{if } s' = R_{35}, \\ \{r_{38}\} & \text{if } s' = R_{38}, \\ \{u\} & \text{if } s' = R_i, i \in \{2, 13, 15, 26, 47\}, \\ \emptyset & \text{otherwise.} \end{cases}$$

In this paper, an MDP is used as an environment with model-free design. In the elements of Definition 2.1, S , A , s_{init} and AP are known, while P and L are unknown.

2.2. Linear Temporal Logic and Büchi Automata

In our MARL method, we express a control specification for a given surveillance system using an LTL formula.

LTL formulas consist of a set of atomic propositions, Boolean operators, and temporal operators. LTL syntax is defined recursively over a set of atomic propositions AP . The syntax is given by the following grammar:

$$\varphi := \top \mid \alpha \in AP \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \mathbf{X}\varphi \mid \varphi_1 \mathbf{U}\varphi_2,$$

where φ , φ_1 , and φ_2 are LTL formulas, $\top$ (true) is a Boolean constant, and $\wedge$ (conjunction) and $\neg$ (negation) are Boolean connectives. For temporal operators $\mathbf{X}$ (next) and $\mathbf{U}$ (until), intuitively, the LTL formula $\mathbf{X}\varphi$ means that φ becomes true at the next time, while the LTL formula $\varphi_1 \mathbf{U}\varphi_2$ means that φ_1 remains true until φ_2 becomes true at some time. The operator

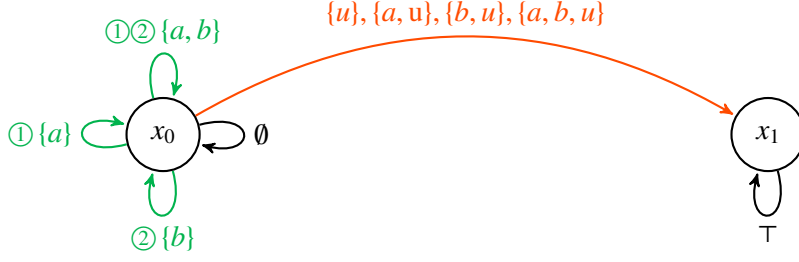


Fig. 2. tLDGBA B_φ converted from the LTL formula $\varphi = \mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{G}\neg u$, where $X = \{x_0, x_1\} = X_{final}$ and $x_{init} = x_0$. Green arcs labeled ① and ② are accepting transitions. Once u is input, B_φ is no longer accepted.

Definition 2.4. A tLDGBA is a tGBA $B = (X, x_{init}, \Sigma, \delta, \mathcal{F})$ where X can be partitioned into two disjoint subsets $X_{initial}$ and X_{final} as follows:

- $F_j \subset X_{final} \times \Sigma \times X_{final}, \forall F_j \in \mathcal{F}$,
- $|\{(x, \sigma, x') \in \delta; \sigma \in \Sigma, x' \in X_{final}\}| = 1, \forall x \in X_{final}$,
- $|\{(x, \sigma, x') \in \delta; \sigma \in \Sigma, x' \in X_{initial}\}| = 0, \forall x \in X_{final}$,
- $\forall (x, \varepsilon, x') \in \delta, x \in X_{initial} \wedge x' \in X_{final}$.

Note that ε -transitions occur only in the transitions from $X_{initial}$ to X_{final} where the empty word ε is the input alphabet. Except for this ε -transition, the transitions occurring in $X_{initial}$ and X_{final} are deterministic, respectively. Hereafter, let B_φ denote a tLDGBA B converted from an LTL formula φ , where an input alphabet is given by $\Sigma = 2^{AP} \cup \{\varepsilon\}$.

We show below a simple example of an LTL formula and a tLDGBA.

Example 2.5. Let us describe the LTL formula φ as follows:

$$\varphi = \mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{G}\neg u,$$

where a, b, u are atomic propositions. By converting this formula φ , we can obtain the tLDGBA B_φ shown in Fig. 2. The acceptance condition of B_φ is $\mathcal{F} = \{F_j\}_{j=1}^2$, where $F_1 = \{(x_0, \{a\}, x_0), (x_0, \{a, b\}, x_0)\}$ and $F_2 = \{(x_0, \{b\}, x_0), (x_0, \{a, b\}, x_0)\}$. In Fig. 2, the green arcs labeled ① and ② represent F_1 and F_2 , respectively. Once u is input, the state of B_φ transitions to x_1 and B_φ is no longer accepted.

For the surveillance problem shown in Example 2.2, we present an example of an LTL formula.

Example 2.6. Consider the surveillance problem shown in Example 2.2 again. We give the mobile robots the following control specification, “to visit the surveillance rooms $R_i (i \in \{4, 23, 35, 38\})$ infinitely often, while to avoid visiting the unsafe rooms $R_i (i \in \{2, 13, 15, 26, 47\})$ ”. We can describe this specification as the following LTL formula φ :

$$\varphi = \mathbf{GFr}_4 \wedge \mathbf{GFr}_{23} \wedge \mathbf{GFr}_{35} \wedge \mathbf{GFr}_{38} \wedge \mathbf{G}\neg u,$$

where $r_i (i = 4, 23, 35, 38)$ are atomic propositions that implies “to visit the rooms $R_i (i = 4, 23, 35, 38)$ ” and u is the proposition that implies “to visit the rooms $R_i (i = 2, 13, 15, 26, 47)$ ”.

In our MARL method, we use an automaton obtained by augmenting a tLDGBA B_φ , which

we call augmented tLDGBA and denote by $\bar{B}_\varphi$. This augmented tLDGBA $\bar{B}_\varphi$ accepts the same language as B_φ and can ensure that accepting transitions occur infinitely often. For details on how a tLDGBA is augmented, see [13].

3. Multi-Agent Reinforcement Learning Method

In this section, we summarize our previous MARL method [12].

Consider N agents tasked with a common goal, i.e., a shared control specification described by an LTL formula. Each agent $i \in \mathcal{N} = \{1, \dots, N\}$ has its own environment and cannot observe each other's state. In order to enable agents to learn cooperative actions under separate environments, we introduce an aggregator. This aggregator calculates how well the shared control specification is satisfied from their actions, and gives each agent a reward according to their actions. In other words, each agent improves its own action and learns to cooperate based on the rewards from the aggregator. To make learning efficient, we assume that each state transition of N agents that occurs stochastically is synchronous, i.e., all agents in $\mathcal{N}$ either move in their intended directions or stay in the same states. To realize this synchronous move, we assume that agents can communicate with others. Moreover, we do not consider any interference of the agents with each other's actions, such as collisions. The collision avoidance functions provided to the agents are described in the numerical experiment in Section 5.

First, each agent $i \in \mathcal{N}$ has a labeled MDP M_i , which represents a given environment. Next, N agents are given a shared control specification, which is described by an LTL formula φ . In addition, by taking the composite product of M_i and (shared) $\bar{B}_\varphi$ for each agent, a product MDP can be constructed. We obtain a control policy by RL on this product MDP. Here, this product MDP utilizes an MDP framework with exogenous inputs and considers information from other agents such as their states, actions, and labels as part of this exogenous input mechanism. Let C_i denote the members of agent group $\mathcal{N}$, excluding agent i , i.e., $C_i := \mathcal{N} \setminus \{i\}$.

A product MDP given for each agent is defined as follows.

Definition 3.1. For each agent $i \in \mathcal{N} = \{1, \dots, N\}$, given an MDP $M_i = (S_i, A_i, s_{init,i}, P_i, AP_i, L_i)$ and an augmented tLDGBA $\bar{B}_\varphi = (\bar{X}, \bar{x}_{init}, \bar{\Sigma}, \bar{\delta}, \bar{\mathcal{F}})$, a product MDP is a tuple $M_i^\otimes = M_i \otimes \bar{B}_\varphi = (S_i^\otimes, A_i^\otimes, s_{init,i}^\otimes, P_i^\otimes, \mathcal{F}_i^\otimes)$. Each element of $M_i^\otimes$ is as follows:

- $S_i^\otimes = S_i \times \bar{X}$: a finite set of states,
- $A_i^\otimes = A_i \cup \{\varepsilon_{\bar{x}}; \exists \bar{x}' \in \bar{X} \text{ s.t. } (\bar{x}, \varepsilon, \bar{x}') \in \bar{\delta}\}$: a finite set of actions, where $\varepsilon_{\bar{x}}$ is a action corresponding to the ε -transition to $\bar{x}' \in \bar{X}$ on $\bar{B}_\varphi$,
- $s_{init,i}^\otimes = (s_{init,i}, \bar{x}_{init})$: an initial state,
- $P_i^\otimes : S_i^\otimes \times A_i^\otimes \times S_i^\otimes \rightarrow [0, 1]$: a transition function, which is represented as follows:

$$P_i^\otimes(s_i^{\otimes'} | s_i^\otimes, a_i^\otimes) = \begin{cases} P_i(s_i' | s_i, a_i) & \text{if } a_i \in \mathcal{A}_i(s_i), (\bar{x}, L_i((s_i, a_i, s_i')) \cup l_{C_i}, \bar{x}') \in \bar{\delta}, \\ & \forall k \in C_i, a_k \in \mathcal{A}_k(s_k), \\ 1 & \text{if } s_i = s_i', a_i^\otimes = \varepsilon_{\bar{x}'}, (\bar{x}, \varepsilon, \bar{x}') \in \bar{\delta}, \\ & \forall k \in C_i, a_k^\otimes = \varepsilon_{\bar{x}'}, \\ 0 & \text{otherwise,} \end{cases}$$

where for each $i \in \mathcal{N}$, $s_i^\otimes = (s_i, \bar{x})$, $s_i^{\otimes'} = (s_i', \bar{x}')$, and $a_i^\otimes = a_i$ or $\varepsilon_{\bar{x}}$. Also, $l_{C_i} = \bigcup_{k \in C_i} (L_k((s_k, a_k, s_k')))$.

- $\mathcal{F}_i^\otimes = \{F_{i,j}^\otimes\}_{j=1}^n$: an accepting condition, where for each $j \in \{1, \dots, n\}$, a set of ac-

cepting transitions is $F_{i,j}^\otimes = \{(s_i^\otimes, a_i^\otimes, s_i^{\otimes'}) \in S_i^\otimes \times A_i^\otimes \times S_i^\otimes; P_i^\otimes(s_i^{\otimes'} | s_i^\otimes, a_i^\otimes) > 0, (\bar{x}, L_i((s_i, a_i, s_i'))) \cup l_{C_i}, \bar{x}') \in \bar{F}_j\}$. where for each $i \in \mathcal{N}$, $s_i^\otimes = (s_i, \bar{x})$, $s_i^{\otimes'} = (s_i', \bar{x}')$, and $a_i^\otimes = a_i$ or $\varepsilon_{\bar{x}'}$. Also, $l_{C_i} = \bigcup_{k \in C_i} (L_k((s_k, a_k, s_k')))$.

Remark 1. Note that Definition 3.1, unlike the general MDP framework, considers exogenous inputs from other agents, since the augmented tLDGBA $\bar{B}_\varphi$ is shared among agents. This structure retains the Markov property within its design. Related works include distributed MDPs [24], which are individualized for each agent and account for interactions among multiple agents. In addition, there are existing works that integrate exogenous inputs within the MDP framework, such as input-driven MDPs [25].

The reward function is defined as a function that gives a reward for each agent if at least one agent causes an accepting transition.

Definition 3.2. Consider a set of agents $\mathcal{N} = \{1, \dots, N\}$, the reward function $\mathcal{R} : S_1^\otimes \times \dots \times S_N^\otimes \times A_1^\otimes \times \dots \times A_N^\otimes \times S_1^\otimes \times \dots \times S_N^\otimes \rightarrow \mathbb{R}_{\geq 0}^N$ is defined as:

$$\mathcal{R}(s_1^\otimes, \dots, s_N^\otimes, a_1^\otimes, \dots, a_N^\otimes, s_1^{\otimes'}, \dots, s_N^{\otimes'}) = \begin{cases} (r_1, \dots, r_N) & \text{if } \exists i \in \mathcal{N}, \exists j \in \{1, \dots, n\}, \\ & (s_i^\otimes, a_i^\otimes, s_i^{\otimes'}) \in \bar{F}_{i,j}, \\ (0, \dots, 0) & \text{otherwise,} \end{cases} \quad (1)$$

where $r_1, \dots, r_N$ is a positive value.

Note that in Definition 3.2, each agent can be given a different reward value.

In our proposed method, we utilize Q-learning [26], which is a well-known reinforcement learning algorithm. The goal is to determine the optimal policy to maximize cumulative rewards. The objective function can be expressed as follows:

$$\mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_{i,t}^\otimes, a_{i,t}^\otimes, s_{i,t}^{\otimes'}) \right], \quad s_{i,0}^\otimes \in S_i^\otimes, \quad a_{i,0}^\otimes \in A_i^\otimes$$

where $\gamma \in [0, 1)$ is the discount factor. In Q-learning, each agent $i \in \mathcal{N}$ has a table-type state-action value function, called a Q-table. The update equation for the Q-table $Q_i(s_i^\otimes, a_i^\otimes)$ is as follows:

$$Q_i(s_i^\otimes, a_i^\otimes) \leftarrow (1 - \alpha)Q_i(s_i^\otimes, a_i^\otimes) + \alpha [r_i + \gamma \max_{a_i^{\otimes'}} Q_i(s_i^{\otimes'}, a_i^{\otimes'})], \quad (2)$$

where $s_i^\otimes \in S_i^\otimes$ is a current state, $a_i^\otimes \in A_i^\otimes$ is a selected action, $s_i^{\otimes'} \in S_i^\otimes$ is the next state, r_i is the positive reward given by the reward function $\mathcal{R}$, $\alpha \in [0, 1]$ is the learning rate.

In this method, we introduce an aggregator that computes the tLDGBA's state transitions and gives rewards to the agents. The learning procedure with the aggregator is described below.

Learning Procedure of Distributed Method:

Step 1: Each agent $i \in \mathcal{N}$ selects an action a_i in the current state s_i on its own MDP M_i based on its policy.

Step 2: Each agent i transitions to the next state s_i' , occurring synchronously through communication among the other agents.

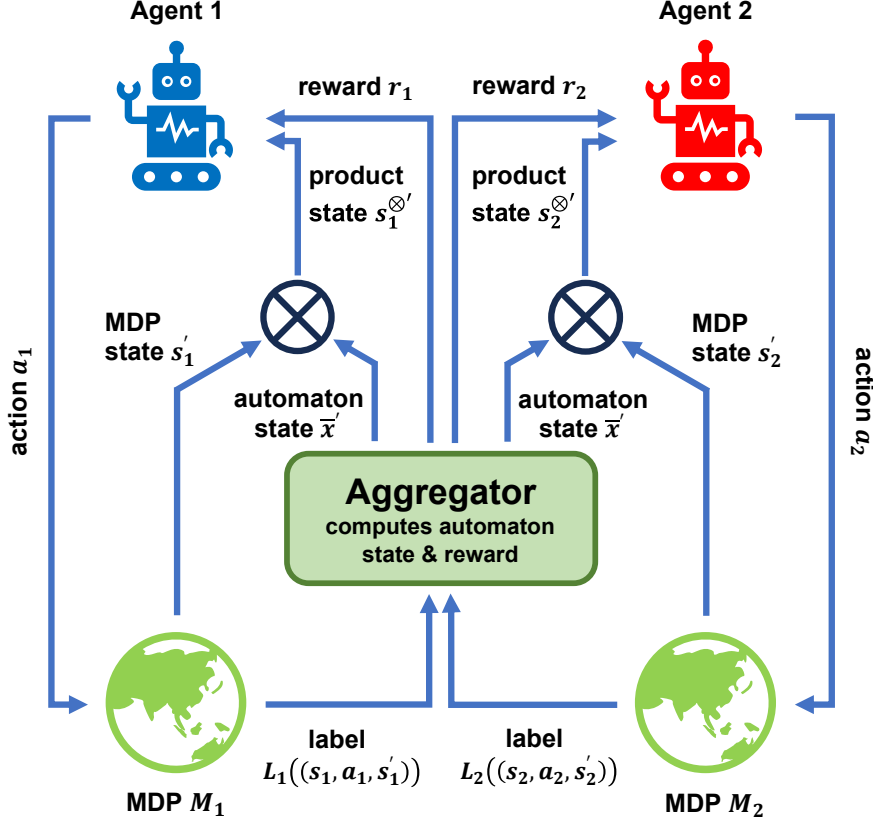


Fig. 3. Learning procedure of our distributed MARL method for two agents ($N = 2$). Note that an MDP state s_i is individual for each agent, while an augmented tLDGBA state $\bar{x}$ is shared. In this learning method, agents are rewarded if an augmented tLDGBA's accepting transition is satisfied due to actions of the agents on an MDP. It is an aggregator that computes these accepting transitions and rewards given to the agents.

- Step 3:** Each labeling function L_i outputs a label, meaning what action each agent i has taken.
- Step 4:** The aggregator collects the union set of the labels and computes the state transition of the (shared) augmented tLDGBA $\bar{B}_\varphi$ to determine the next state $\bar{x}'$.
- Step 5:** By combining s_i' and $\bar{x}'$, each next state $s_i^{\otimes'}$ on the product MDP $M_i^\otimes$ is determined, which is received by each agent i .
- Step 6:** The aggregator (the reward function $\mathcal{R}$) computes and distributes each agent i 's reward r_i .
- Step 7:** Each agent updates its own Q-table $Q_i(s_i^\otimes, a_i^\otimes)$. Return to Step 1 and repeat the above procedure.

Fig. 3 shows the learning procedure of this method in the case of two agents ($N = 2$).

4. Reward Distribution Method for Multi-Agent Surveillance Systems

In our MARL method introduced in Section 3, an aggregator computes rewards given to agents depending on whether the accepting transitions are satisfied or not. From Definition 3.2, values of the rewards given by the reward function $\mathcal{R}$ may be different. Also, from (2), the update equation of the Q-table exists for each agent, and the learning can be differentiated

by varying each reward. Focusing on these points, we consider varying the amount of reward given to each agent according to their contribution.

In this section, we describe how to evaluate the contribution of each agent in a multi-agent surveillance system. We then propose a reward distribution method and describe some cases of reward design.

In this study, to calculate the contribution that determines the amount of reward distribution, we use the distance between each agent and each surveillance room. A smaller distance is considered as a higher contribution and a higher reward is given to the agent. This reward distribution method is expected to improve learning performance.

4.1. Definition of Some Symbols

To design a reward, we define some symbols. Let $\mathcal{N}_{acc} \subseteq \mathcal{N}$ denote a set of agents satisfying an accepting transition. Let K denote a set of surplus surveillance rooms other than the rooms in which the accepting transition occurs. For each agent $j \in \mathcal{N} \setminus \mathcal{N}_{acc}$, let $\text{dist}(s_j, k)$ denote the distance between the current MDP state $s_j \in S_j$ and the surveillance room $k \in K$, where the difference set $\mathcal{N} \setminus \mathcal{N}_{acc}$ means a set of agents not satisfying an accepting transition. Furthermore, the shortest distance, i.e., the distance to the nearest surveillance room is denoted by

$$d_j = \min_{k \in K} \text{dist}(s_j, k).$$

Here, let $k_{\min, j}$ denote the nearest surveillance room for agent $j \in \mathcal{N} \setminus \mathcal{N}_{acc}$. Consider the case in which $k_{\min, j}$ is competitive among agents, i.e., there exist simultaneously agents j and j' such as $k_{\min, j} = k_{\min, j'}$. Assume that by comparing the shortest distances $\text{dist}(s_j, k_{\min, j})$ and $\text{dist}(s_{j'}, k_{\min, j'})$, the agent with the smaller value is rewarded, and the other is not. In such a case, the rewarded agent j^* can be expressed as follows:

$$j^* = \arg \min_{j \in \mathcal{N} \setminus \mathcal{N}_{acc}} \text{dist}(s_j, k_{\min, j}).$$

4.2. Reward Design

Given the maximum reward $r_{acc} \in \mathbb{R}_{\geq 0}$ for agents in $\mathcal{N}_{acc}$ and the shortest distance d_j derived from the surveillance system, a reward distributing function $f(r_{acc}, d_j)$, which computes a reward according to d_j for each agent $j \in \mathcal{N} \setminus \mathcal{N}_{acc}$, takes the value range shown below:

$$0 \leq f(r_{acc}, d_j) \leq r_{acc}.$$

Thus, from Definition 3.2, an individual reward r_i given to agent $i \in \mathcal{N}$ can be expressed as follows:

$$r_i = \begin{cases} r_{acc} & \text{if } i \in \mathcal{N}_{acc}, \\ f(r_{acc}, d_i) & \text{otherwise.} \end{cases}$$

Based on the above definition in Section 4.1, we propose a reward design method. We suppose that agents are rewarded based on the shortest distances to the surveillance room.

Then, the function $f(r_{acc}, d_j)$ is given by

$$f(r_{acc}, d_j) = \begin{cases} r_{acc}/g(d_j) & \text{if } j = j^*, \\ 0 & \text{otherwise,} \end{cases}$$

where $g(d_j)$ is a monotonically increasing function that takes positive values. In this proposed method, for agents that do not satisfy an accepting transition, the contribution is calculated according to their distance to the nearest surveillance room. The closer the distance, the larger the value of $g(d_j)$, and thus the larger their reward. Therefore, it is expected that the agents will take more appropriate routes, reducing the unnecessary actions that have nothing to do with patrolling.

Thus, based on the symbols defined, we can consider reward design according to the contribution of each agent.

We represent an example of a reward distribution.

Example 4.1. Consider one case of reward distribution for three mobile robots shown in Fig. 4.

If an accepting transition occurs for at least one agent, the other agents are rewarded according to their distance from themselves to the nearest surveillance room. In the case of Fig. 4, agent 1 that causes the accepting transition receives the maximum reward r_{acc} . Whereas, agents 2 and 3 receive the reward $f(r_{acc}, 2)$ and $f(r_{acc}, 3)$, since the shortest distance to the nearest surveillance room is 2 and 3 steps, respectively.

In the other case of Fig. 5, agent 1 receives the same reward r_{acc} as before after the acceptance transition. Whereas, agents 2 and 3 are competing for the same room, $k_{min,2} = k_{min,3}$. In this case, agent 2 receives the reward $f(r_{acc}, 2)$ because agent 2 has a shorter distance ($\text{dist}(s_2, k_{min,2}) < \text{dist}(s_3, k_{min,3})$), while agent 3's reward is $(r_{acc}, 3) = 0$.

5. Numerical Experiment

To verify the effectiveness of the proposed reward distribution method, we experiment with the three reward design cases for a multi-agent surveillance system. Within one of these design cases, we include the method of our previous work [12] for comparison with the proposed method.

Consider a surveillance problem for three mobile robots ($N = 3$) patrolling the 7×7 grid world shown in Fig. 1. In this problem, we assume that the mobile robots are equipped with a collision avoidance function. It is also assumed that each room in the grid world is large enough for the mobile robots and that they can move past each other without overlapping. We apply our MARL method and reward distribution method to the system with the MDP given in Example 2.2, the LTL formula given in Example 2.6. Consider the following three cases of reward design:

Case 1: Design without consideration of reward distribution [12]

We suppose that all agents obtain the same reward:

$$f(r_{acc}, d_j) = r_{acc}.$$

This design is the same as in [12].

Case 2: Design to distribute at a uniform discount

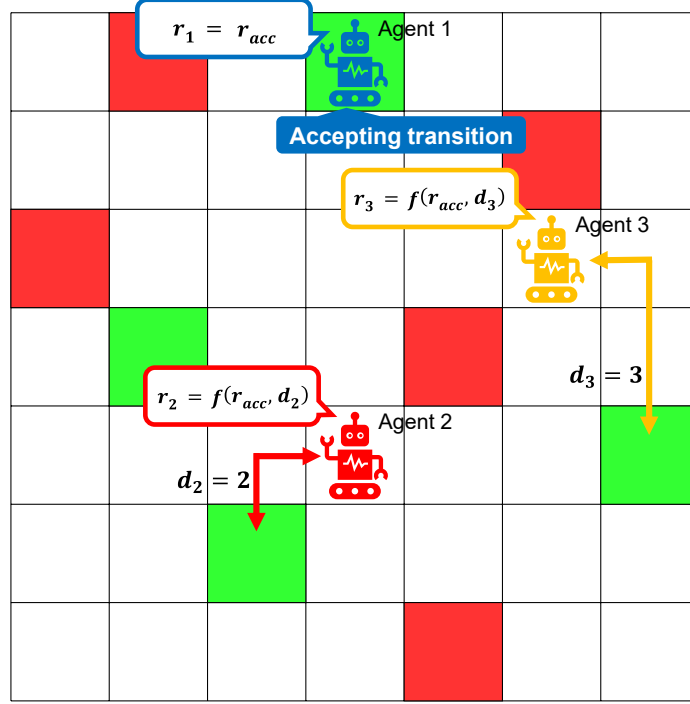


Fig. 4. Example of reward distribution for three agents ($N = 3$). Agent 1 satisfying accepting transition receives the maximum reward r_{acc} , while agents 2 and 3 receive the reward $f(r_{acc}, d_2)$ and $f(r_{acc}, d_3)$ according to the distance to the nearest surveillance room.

We suppose that agents are rewarded at the same value, which is uniformly discounted:

$$f(r_{acc}, d_j) = \begin{cases} r_{acc}/c & \text{if } j = j^*, \\ 0 & \text{otherwise,} \end{cases}$$

where c is a positive constant.

Case 3: Proposed Method (Design considering the shortest distances to the surveillance room)

In Case 1, the contribution is evaluated the same for all agents, regardless of whether they take an accepting transition or not. Even agents whose actions are not related to patrolling surveillance rooms are equally rewarded. It is possible that this has a negative impact on their learning of cooperative actions. In Case 2, the contribution of the agent that does not satisfy an accepting transition is evaluated lower than that of the agent that does. This allows them to learn that accepting transitions are the better actions for them. However, the amount of reward given to them does not depend on their distance from the surveillance room, and they can learn the difference only if they experience accepting transitions. Therefore, it is possible that there will be cases in which some agents cannot learn the behavior of visiting surveillance rooms.

We ran a total of 500 learning sessions of 2000 episodes, with 2000 steps in each episode. The learning parameters are that the discount rate is $\gamma = 0.99$ and the learning rate is $\alpha = 1/(1 + ep)^{0.5}$, where ep is the number of episodes. Also, the ε -greedy method is used for each agent's policy, with $\varepsilon = 0.99^{ep}$. In reward design, we set the reward for agents $r_{acc} = 2$, the positive constant $c = 5$ for Case 2, and a monotonically increasing $g(d_j) = (0.3 + d_j)^3$ for

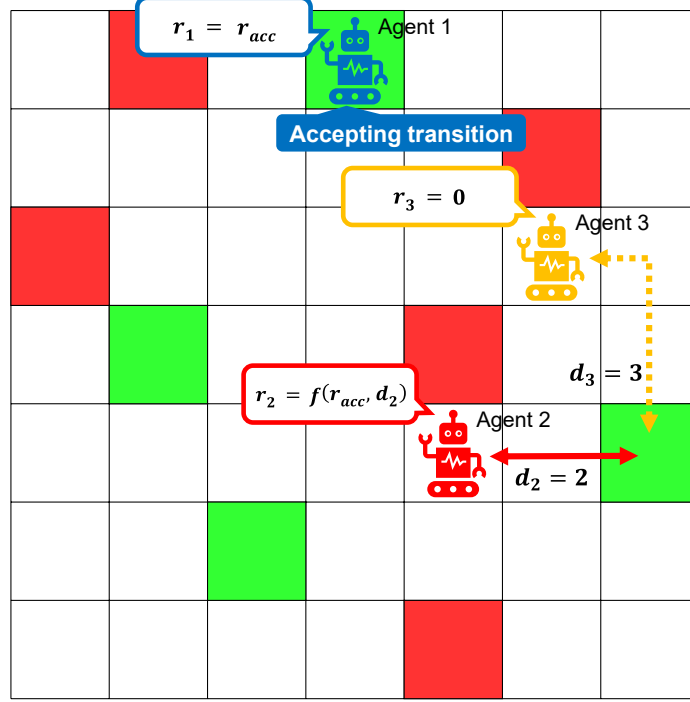


Fig. 5. Example of reward distribution for three agents ($N = 3$) when the same surveillance room is competitive between agents. Agent 1 satisfying accepting transition receives the maximum reward r_{acc} , while agents 2 and 3 are competing for the same room. Comparing the shortest distance of each agent, agent 2 receives the reward $f(r_{acc}, d_2)$ and agent 3 receives no reward.

Case 3. The group of mobile robots learns to cooperate by repeatedly patrolling the green rooms R_i ($i = 4, 23, 35, 38$), sharing the responsibility within the group, while avoiding the red rooms R_i ($i = 2, 13, 15, 26, 47$).

Figs. 6–8 show the reward history obtained by each reward design case. In these reward histories, the mean and standard deviations of average rewards at the end of the learning are $1.91(\pm 0.64)$ (Case 1), $2.14(\pm 0.79)$ (Case 2), and $2.33(\pm 0.79)$ (Case 3). As shown in Figs. 6–8 and the above values, we see that the obtained rewards are increased in the design where the reward distribution is based on the shortest distance.

Furthermore, Fig. 9 presents a sample path of three mobile robots resulting over 500 learning sessions, with circular markers in the figure denoting accepting transitions. This figure illustrates the periodic behavior that satisfies the given LTL specification. Across these sessions, the percentages of successful results satisfying the LTL specification are as follows: 97.8% for Case 1, 96.0% for Case 2, and 98.4% for Case 3. Thus, in all Cases 1–3, we see that most of results satisfy the LTL specification.

6. Conclusion and Future Work

In this paper, we proposed a reward distribution method in cooperative MARL for a surveillance system with an LTL specification. Through the numerical experiment, we showed that the performance was improved by the reward design considering the shortest distance to the surveillance rooms.

Future works include further increasing the amount of reward by devising the distribution method. In addition, although this paper assumed a surveillance problem, we will consider more general distribution methods, for example, incorporating MDP, LTL, or tLDGBA into

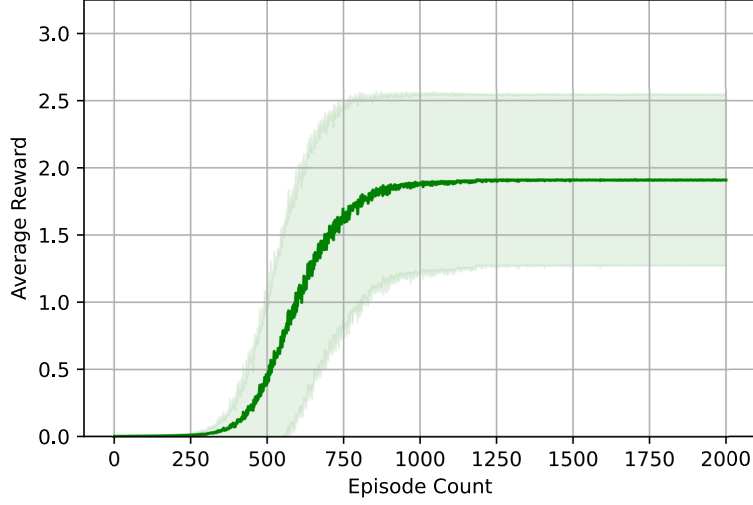


Fig. 6. Reward history in Case 1: Design without consideration of reward distribution. The mean of the average reward for each episode was calculated across 500 learning sessions. The green areas indicate the range of standard deviations.

the contribution calculation. Moreover, in the future, we will delve into more complex LTL specifications, not only simple expressions using only G and F as presented in this paper. For example, there may be cases where additional time operators such as X and U are incorporated into the LTL expression, or where specifications dealing with interactions between agents are incorporated. By addressing these complexities, we will demonstrate the applicability of our devised method across diverse and challenging scenarios.

This work was partly supported by JSPS KAKENHI Grant Numbers JP21H04558, JP22K04163, JP23H01430.

References

- [1] Weiss G. Multiagent systems: a modern approach to distributed artificial intelligence. MIT press; 1999.
- [2] Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT press; 2018.
- [3] Baier C, Katoen JP. Principles of model checking. MIT press; 2008.
- [4] León BG, Belardinelli F. Extended markov games to learn multiple tasks in multi-agent reinforcement learning. arXiv preprint arXiv:200206000. 2020;.
- [5] Sun C, Li X, Belta CA. Automata guided semi-decentralized multi-agent reinforcement learning. 2020 American Control Conference. 2020;:3900–3905.
- [6] Hammond L, Abate A, Gutierrez J, et al. Multi-agent reinforcement learning with temporal logic specifications. arXiv preprint arXiv:210200582. 2021;.
- [7] Wang J, Yang S, An Z, et al. Multi-agent reinforcement learning guided by signal temporal logic specifications. arXiv preprint arXiv:230606808. 2023;.
- [8] Saha I, Ramaithitima R, Kumar V, et al. Automated composition of motion primitives for multi-robot systems from safe ltl specifications. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems; IEEE; 2014. p. 1525–1532.
- [9] Tumova J, Dimarogonas DV. Multi-agent planning under local ltl specifications and event-based synchronization. Automatica. 2016;70:239–248.
- [10] Loizou SG, Kyriakopoulos KJ. Automated planning of motion tasks for multi-robot systems. In: Proceedings of the 44th IEEE Conference on Decision and Control; IEEE; 2005. p. 78–83.

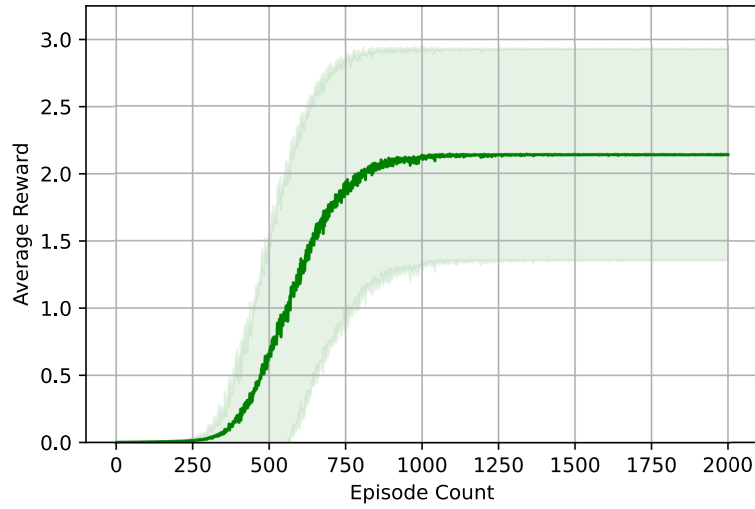


Fig. 7. Reward history in Case 2: Design to distribute at a uniform discount. The mean of the average reward for each episode was calculated across 500 learning sessions. The green areas indicate the range of standard deviations.

- [11] Kloetzer M, Ding XC, Belta C. Multi-robot deployment from ltl specifications with reduced communication. In: 2011 50th IEEE Conference on Decision and Control and European Control Conference; IEEE; 2011. p. 4867–4872.
- [12] Terashima K, Kobayashi K, Yamashita Y. Control of multi-agent surveillance systems using reinforcement learning. In: SICE Annual Conference 2022; 2022. p. 690–693.
- [13] Oura R, Sakakibara A, Ushio T. Reinforcement learning of control policy for linear temporal logic specifications using limit-deterministic generalized Büchi automata. *IEEE Control Systems Letters*. 2020;4(3):761–766.
- [14] Foerster J, Assael IA, De Freitas N, et al. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*. 2016;29.
- [15] Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*. 2017;.
- [16] Foerster J, Farquhar G, Afouras T, et al. Counterfactual multi-agent policy gradients. In: *Proceedings of the AAAI conference on artificial intelligence*; Vol. 32; 2018.
- [17] Wang J, Ren Z, Liu T, et al. Qplex: Duplex dueling multi-agent Q-learning. *arXiv preprint arXiv:2008.01062*. 2020;.
- [18] Lowe R, Wu YI, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*. 2017;30.
- [19] Sombolestan S, Rasooli A, Khodaygan S. Optimal path-planning for mobile robots to find a hidden target in an unknown environment based on machine learning. *Journal of ambient intelligence and humanized computing*. 2019;10:1841–1850.
- [20] Qie H, Shi D, Shen T, et al. Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning. *IEEE access*. 2019;7:146264–146272.
- [21] Setyawan GE, Hartono P, Sawada H. Cooperative multi-robot hierarchical reinforcement learning. *Int J Adv Comput Sci Appl*. 2022;13:35–44.
- [22] Büchi JR. On a decision method in restricted second order arithmetic. In: Nagel E, Suppes P, Tarski A, editors. *International Congress on Logic, Methodology and Philosophy of Science*. Stanford University Press; 1962. p. 1–11.
- [23] Rabin MO. Decidability of second-order theories and automata on infinite trees. *Transactions of the american Mathematical Society*. 1969;141:1–35.
- [24] Park H, Van Der Schaar M. A framework for foresighted resource reciprocation in p2p networks. *IEEE Transactions on Multimedia*. 2008;11(1):101–116.

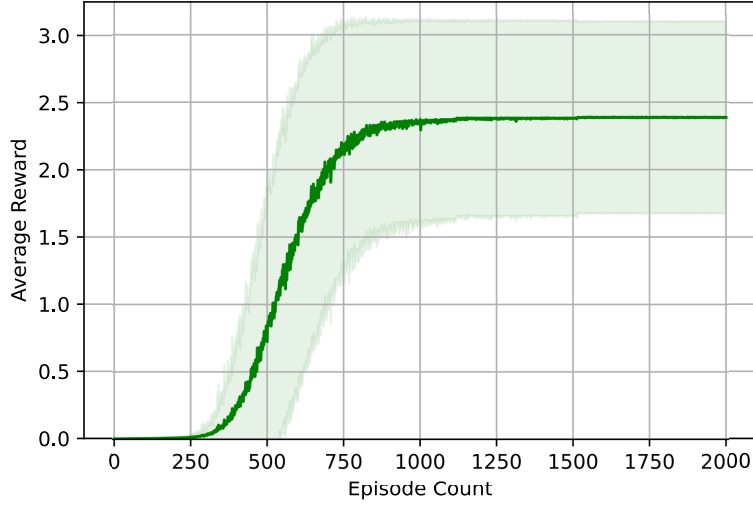


Fig. 8. Reward history in Case 3: Design considering the shortest distances to the surveillance room. The mean of the average reward for each episode was calculated across 500 learning sessions. The green areas indicate the range of standard deviations.

- [25] Mao H, Venkatakrishnan SB, Schwarzkopf M, et al. Variance reduction for reinforcement learning in input-driven environments. arXiv preprint arXiv:180702264. 2018;.
- [26] Watkins CJ, Dayan P. Q-learning. Machine learning. 1992;8:279–292.

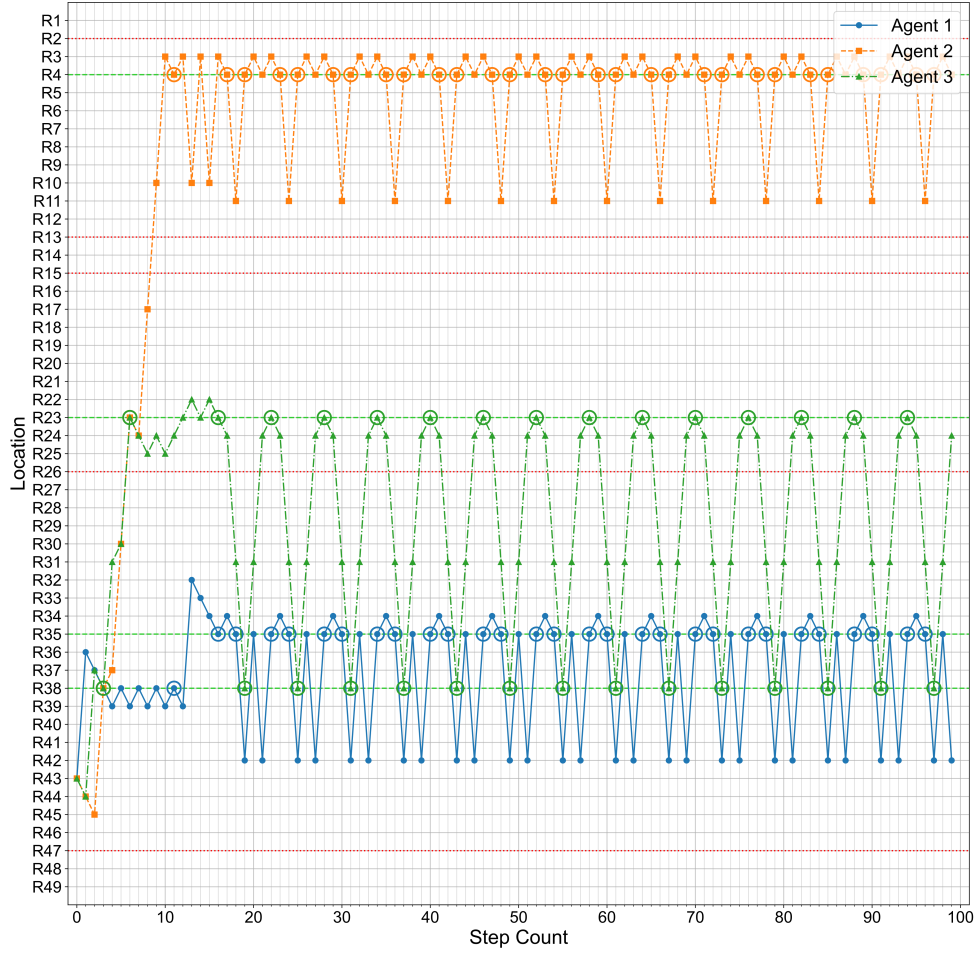


Fig. 9. Sample path of three mobile robots resulting from training. R_i ($i \in \{4, 23, 35, 38\}$) are surveillance rooms, while R_i ($i \in \{2, 13, 15, 26, 47\}$) are unsafe rooms. The circular markers represent accepting transitions.