big data and cognitive computing

MDPI

Article

Runtime Verification-Based Safe MARL for Optimized Safety Policy Generation for Multi-Robot Systems

Yang Liu * and Jiankun Li

Institute of Logistics Science and Engineering, Shanghai Maritime University, Shanghai 200120, China; 202130510003@stu.shmtu.edu.cn

* Correspondence: lyang@shmtu.edu.cn

Abstract: The intelligent warehouse is a modern logistics management system that uses technologies like the Internet of Things, robots, and artificial intelligence to realize automated management and optimize warehousing operations. The multi-robot system (MRS) is an important carrier for implementing an intelligent warehouse, which completes various tasks in the warehouse through cooperation and coordination between robots. As an extension of reinforcement learning and a kind of swarm intelligence, MARL (multi-agent reinforcement learning) can effectively create the multi-robot systems in intelligent warehouses. However, MARL-based multi-robot systems in intelligent warehouses face serious safety issues, such as collisions, conflicts, and congestion. To deal with these issues, this paper proposes a safe MARL method based on runtime verification, i.e., an optimized safety policy-generation framework, for multi-robot systems in intelligent warehouses. The framework consists of three stages. In the first stage, a runtime model SCMG (safety-constrained Markov Game) is defined for the multi-robot system at runtime in the intelligent warehouse. In the second stage, rPATL (probabilistic alternating-time temporal logic with rewards) is used to express safety properties, and SCMG is cyclically verified and refined through runtime verification (RV) to ensure safety. This stage guarantees the safety of robots' behaviors before training. In the third stage, the verified SCMG guides SCPO (safety-constrained policy optimization) to obtain an optimized safety policy for robots. Finally, a multi-robot warehouse (RWARE) scenario is used for experimental evaluation. The results show that the policy obtained by our framework is safer than existing frameworks and includes a certain degree of optimization.

Keywords: intelligent warehouse; multi-robot systems; multi-agent reinforcement learning; runtime verification

Citation: Liu, Y.; Li, J. Runtime Verification-Based Safe MARL for Optimized Safety Policy Generation for Multi-Robot Systems. *Big Data Cogn. Comput.* **2024**, *8*, 49. https://doi.org/10.3390/bdcc8050049

Academic Editors: Robert Ross and Alex Stumpf

Received: 7 April 2024
Revised: 3 May 2024
Accepted: 13 May 2024
Published: 16 May 2024

1. Introduction

The intelligent warehouse is a logistics-warehousing-management -system platform and an important application scenario of the 5G Industrial Internet. Intelligent warehouses improve the efficiency and accuracy of warehouses through automated warehousing, automated sorting, automated distribution, and other intelligent processes, while also reducing labor costs and enhancing service quality [1]. In addition, intelligent warehouses can optimize warehouse logistics operations and improve business-response speed through real-time data analysis and intelligent decision-making, thereby better meeting customer needs and bringing greater economic benefits and market competitiveness to enterprises. Intelligent warehouses can improve the competitiveness and sustainability of the national economy [2].

Multi-Robot Systems (MRSs) [3] in smart warehouses are a type of technology that uses multiple robots to work together to complete warehouse management and logistical operations. This system combines automation, machine learning, artificial intelligence, and advanced sensing technologies to optimize the efficiency and accuracy of warehouse operations. These robots can be autonomously moving AGVs (automated guided vehicles) [4], automatic stacking robots, drones, etc. They achieve efficient work processes and

precise positioning through communication and coordination with a central or decentralized control system. Multi-robot systems bring new opportunities and challenges to the logistics industry. They have brought many benefits to intelligent warehouses, including increasing cargo loading and unloading speeds, reducing labor costs, reducing operational risks, and accurately tracking the location and status of cargo [5].

As a paradigm of swarm intelligence, multi-agent reinforcement learning (MARL) [6] extends reinforcement learning (RL) [7]. with single-agent to multi-agent scenarios [8]. MARL has been widely used in various fields such as industrial automation, logistical distribution, rescue missions, etc. Theoretically speaking, MARL can effectively organize the multi-robot systems in intelligent warehouses. However, the randomness of MARL limits its application in intelligent warehouses multi-robot systems. In the context of MARL, ensuring the safety of the system poses important and complex challenges. These challenges come not only from the design of the algorithm itself, but also from the uncertainty of the environment and the interactions between multiple agents. Although safe multi-agent reinforcement learning is an active research area that has attracted much attention, it still has some shortcomings. For example, current safe MARL algorithms still need to improve their modeling of uncertainty, especially when facing partially observable environments or partially observable agents. Furthermore, existing safe MARL methods are usually designed for specific problems and scenarios, which are difficult to generalize to other fields. The applicability and scalability of these algorithms remain a challenge when facing different types of tasks and environments. In a word, it is difficult to provide formal safety guarantees for MARL. Therefore, MARL-based multi-robot systems in intelligent warehouses face a series of challenges, one of which is how to ensure the safety and reliability of multi-robot operations. How to design safe multi-agent reinforcement learning (safe MARL) [9] to achieve real-time verification, risk identification, and intelligent decision support for multiple robots has become an important topic in multi-robot systems of intelligent warehouses.

This paper aims to explore a universal and formal safety guarantee framework of MARL for multi-robot systems in intelligent warehouses. We exploit runtime verification (RV) [10] to construct a safety-constrained runtime model of a multi-robot system, and propose a safe MARL method, i.e., an optimized safety policy-generation framework, for multi-robot systems in intelligent warehouses, as shown in Figure 1. It includes three stages: Modeling @ Runtime, Runtime Verification, and Constraint Training. In the first stage, we extend the Markov Game (MG) with safety constraints to define a runtime model SCMG (safety-constrained Markov Game) for multi-robot systems in intelligent warehouses. In the second stage, rPATL (probabilistic alternating-time temporal logic with rewards) is used to express safety properties, and SCMG is cyclically verified and refined through runtime verification (RV) to ensure the safety of the model SCMG. This stage guarantees the safety of robot behaviors before training. In the third stage, the verified SCMG guides MARL, i.e., SCPO (safety-constrained policy optimization), to obtain an optimized safety policy for robots. Finally, a multi-robot warehouse (RWARE) scenario of the Shanghai Intelligent Warehouse is used to demonstrate the effectiveness of our framework. Experimental results show that our framework is safer than existing methods and can obtain policies with better performance than general baseline methods. This work will provide useful guidance and inspiration for the development and practical application of multi-robot systems in intelligent warehouses, and will further promote the widespread application of safe MARL.

The remainder of this paper is structured as follows: Section 2 introduces related work on safe RL and safe MARL, explaining the shortcomings of existing work; Section 3 introduces preliminary concepts and terminology; Section 4 introduces our safe MARL framework based on runtime verification, i.e., optimized safety policy-generation framework for multi-robot systems; Section 5 presents the experimental cases and results; and, finally, Section 6 concludes the paper. For full-text abbreviations and their meanings, please see Table A1 in Appendix A.

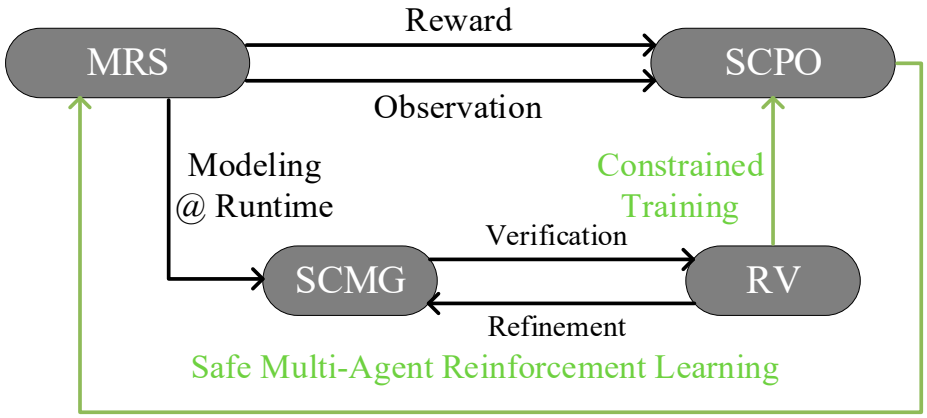

Figure 1. Overview of a universal and formal safety guarantee framework of MARL for MRS.

2. Related Works

In this section, we mainly review two categories of works related to this paper. One is the safety policy of MARL (also known as safe MARL in some references) [11]. The other is the policy optimization of MARL [12]. Comparatively speaking, the former is closer to our work.

2.1. Safety Policy of MARL

Informal methods for specific problems. Traditional safe RL constructs controllers that meet specifications under known environmental dynamics. These methods combine reaction synthesis with precise environmental modeling and abstraction but are computationally too difficult for more complex dynamics [13]. Despite the importance of safe MARL [14], most methods are specifically designed for learning robotic tasks and there is a lack of evaluation of their applicability and generalization capabilities. For example, barrier certificate techniques [15] or model-predictive shielding from control theory [16] are used to model safety. However, these methods are derived from the traditional robotics perspective. CMIX [17] extends QMIX [18] by modifying the reward function to consider peak constraint violations, but this approach has no safety guarantees during training. Some approaches to safe MARL [15] are to reduce unsafe behaviors through human interference [19], but this is not applicable to all problems. Although the above methods take into account the multi-agent safety, they are not applicable to general MARL problems but only focus on specific problems.

Constraint-based informal methods. In safety reinforcement learning, constrained Markov decision processes (CMDPs) [20] are used to ensure that certain safety or performance criteria are met while optimizing the behavior policy to maximize the expected reward. A study conducted in safe RL utilized constrained MDP and proposed a novel policy-optimization algorithm that uses a convex quadratic function obtained from a policy-gradient estimator [21]. Gu et al. [12] proposed using a constrained Markov Game to describe the safe MARL problem and used policy-optimization theory to implement the policy update of safety constraints. However, it is difficult for the above methods to provide formal safety guarantees. This lack of formal guarantees is due to the difficulty in appropriately adjusting the reward-function-customization process [22].

Formal methods. Several recent works [23–25] developed reward-shaping techniques to transform logical constraints expressed in Linear Temporal Logic (LTL) into reward functions for RL. Garcıa et al. [11] considered different safety goals for RL, such as reward variance or limited access to error states. In this work, we synthesize safety specifications that are expressed in LTL [26], a logical system for expressing temporal properties, in particular conditions that a system must satisfy during its execution [27]. For example, LTL has been used to express complex task specifications for robotic planning and control [28]. However, LTL is not sufficient for MARL methods required to learn policies that guarantee safety (e.g., collision-free movement).

Formal verification methods. This technology has been successfully applied to multi-agent systems [29,30] and single-agent reinforcement learning [31]. Such agent-behavior-constraint techniques are well documented in safe RL [12], but they can be overly restrictive, unnecessarily reducing the agent's potential optimality or eliminating behaviors needed to complete a task. Mason et al. [31] proposed a technique to avoid this problem while still providing formal guarantees on the level of safety at which agent learning is considered acceptable. Assured RL [32] took the novel step of incorporating the QV stage into the RL process and produced very promising results. There are recent works [33,34] that apply it to MARL research, but there is still the problem of reducing the optimality of the agent.

2.2. Policy Optimization in MARL

The policy safety described in Section 2.1 is very close to our work. In addition, there is another topic that is also related to this work, which is policy optimization in MARL. We will present some recent advances in this topic.

Policy optimization methods estimate the behavior value of each agent by introducing a baseline policy. They try to directly optimize the policy to maximize the cumulative reward. Witt et al. [35] propose an independent proximal policy optimization (IPPO) method, i.e., a multi-agent variant of proximal policy optimization, which has limitations in cooperative MARL. Yu et al. [36] propose a Multi-Agent Proximal Policy Optimization (MAPPO) method, which can effectively handle training problems in multi-agent environments. For high-dimensional state spaces or complex tasks, it needs more training samples to converge than do other methods. Kuba et al. [37] propose a Heterogeneous-Agent Proximal Policy Optimization (HAPPO) method to solve training problems in heterogeneous-agent environments, which requires more computing resources and training samples. Ye et al. [38] put forward a Shared Network Actor-Critic (SNAC) method based on Actor-Critic MARL. When one agent behaves poorly, it will affect the performance of other agents in the SNAC method. The Independent Actor-Critic (IAC) [39] method is very effective in dealing with problems in continuous action space and high-dimensional-state space but faces problems such as low sample efficiency and unstable training. Christianos et al. [40] propose a Shared Experience Actor-Critic (SEAC) method to improve training effects on agents through shared experience, but shared experience may lead to competition and conflict between agents. In all the above policy optimization methods of MARL, agents may choose unsafe behaviors, which will affect the performance and safety of the entire multi-agent system.

2.3. The Relation between Our Work and Existing Works

Our work belongs to the category of formal verification methods for the safety policy of MARL. It is a universal and formal safety guarantee framework for MARL, based on runtime verification. We first extend the general modeling method of safe MARL to modeling at runtime, and then innovatively use runtime verification to formally ensure the safety of agent behavior before training. Compared with other formal verification methods for MARL, the advantage of our work is that it can ensure a certain degree of optimization based on safety through the constrained training of agents. In summary, our work has made outstanding contributions in terms of scalability, formally ensuring the safety of robot behaviors, and optimizing the performance of multi-robot policies. In addition to this, we apply this framework to the multi-robot system in the Shanghai Intelligent Warehouse, which exhibits good performance.

3. Preliminaries

In this section, we introduce the preliminaries of our work, i.e., reinforcement learning, multi-agent reinforcement learning, Markov Games, and runtime verification.

Reinforcement learning (RL). In RL, the agent seeks to maximize its total reward over time through trial and error. This involves a sequence of observations, actions, and

rewards. The environment provides feedback in the form of rewards, which can be positive or negative, based on the actions taken by the agent. The key components of an RL setup are the agent, the environment, the policy, the reward signal, and the value function. The agent is the decision-maker, the environment includes everything the agent interacts with, the policy is the strategy used by the agent to decide its actions, the reward signal indicates how well the action taken by the agent aligns with the goal, and the value function estimates the long-term benefit of the decisions.

Multi-agent reinforcement learning (MARL). MARL extends the principles of traditional RL to environments where multiple agents interact simultaneously. In MARL, each agent learns to make decisions based on their own experiences, while also considering the actions and potential strategies of other agents. This creates a complex dynamic, as each agent's actions can affect the state of the environment and, consequently, the outcomes for other agents.

The central challenge in MARL is that of coordination versus competition among agents. Agents may either cooperate to achieve a common goal, compete against each other for resources, or exhibit a mixture of both behaviors. The dynamics of these interactions make learning and decision-making significantly more complex than in single-agent scenarios. MARL models typically involve defining policies for each agent that maximize their expected rewards over time. These policies must account for the joint state and action spaces of all agents, which exponentially increases the complexity of the problem. Learning can be centralized, where a single entity observes and controls all agents, or decentralized, where each agent operates independently based on local observations.

Applications of MARL are found in areas such as autonomous vehicle coordination, robotic teams, strategic game playing, and resource management, where multiple decision-makers must effectively interact within a shared environment.

Markov Game (MG). Single-agent reinforcement learning tasks are usually modeled as Markov Decision Processes (MDPs), and Markov Game can be considered a multi-agent extension of MDP. Markov Game, which is also known as random game or Markov countermeasure, is a theoretical framework that combines dynamic programming and game theory. It describes how multiple agents choose strategies in an environment with state transitions to achieve their respective goals. The mathematical analysis of Markov Game usually involves solving the optimal policy, that is, a series of decision rules that define which action should be chosen in each state to maximize the total reward. The Bellman equation is the basis for finding the optimal policy, which expresses the problem of maximizing the sum of the immediate reward obtained by the current action and the expected future reward.

Runtime verification (RV). Runtime verification, in this paper, refers to obtaining system runtime parameters to build a runtime system model and then verifying the safety of the system in the subsequent finite time as efficiently as possible based on model-checking methods. It is a lightweight system-verification technique at runtime, compared with traditional model checking at design time. Runtime verification includes modeling at runtime and verification at runtime, which is complementary to modeling and verification at design time. It is an important component of MBSE (Model-Based Systems Engineering) and can be seen as the lightweight-level MBSE.

We use PRISM-games as the model checker for runtime model of the multi-robot system. PRISM-games provide a formal language and tools that allow users to describe and analyze systems with uncertainty and randomness. It supports many types of game models, including zero-sum games, Markov decision processes (MDP), stochastic games, etc.

4. Optimized Safety Policy-Generation Framework

As shown in Figure 2, the safe MARL method for multi-robot systems, i.e., the optimized safety policy-generation framework, consists of three stages.

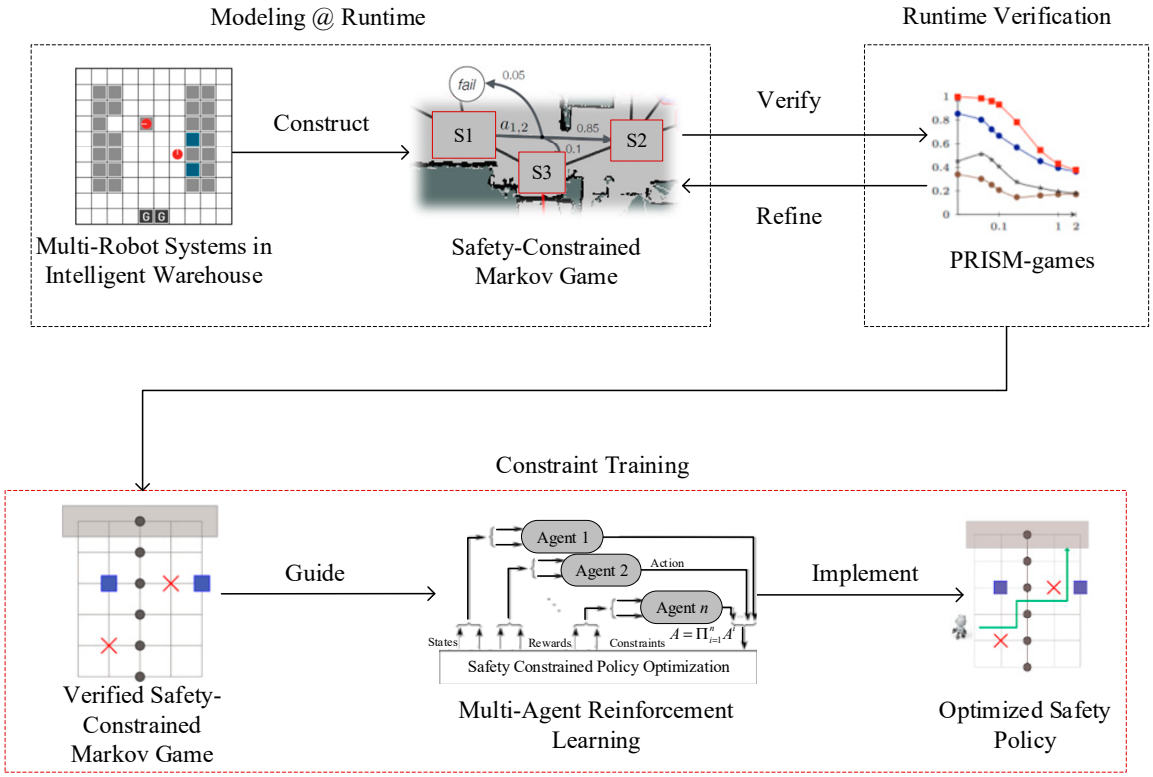

Figure 2. Optimized safety policy-generation framework for multi-robot systems in intelligent warehouses.

In the first stage, we use the runtime information of multi-robot requirements, environmental requirements, and safety requirements to define a runtime model SCMG (Safety-Constrained Markov Game), which is a safety-constraint extension of the Markov Game. At this stage, it is important to remove all unnecessary details of the problem to avoid the state space explosion problem. Therefore, during data collection, we mainly need to identify the following: all potential states of the agent in the environment, the capabilities of the agent, the reward structure needed to evaluate the system, the objectives, and the constraints.

In the second stage, the runtime model SCMG is verified by PRISM-games, and refined until all safety properties are satisfied, in which functional requirements and safety constraints are expressed with rPATL [41]. Runtime verification using PRISM-games ensures that the model SCMG satisfies the safety constraints, i.e., only safe actions can be taken in the multi-robot system being considered. If not, then counterexamples from model checking are used to refine and remove model actions until the safety constraints and functional requirements are met.

The third stage involves generating a safety policy, in which the verified SCMG guides safety-constrained policy optimization (SCPO) to train multi-agents to obtain an optimized safety policy. SCPO uses constrained optimization methods to maximize the objective function and satisfy the constraints, using a technique called Lagrangian Relaxation to solve constrained optimization problems. As the verified SCMG continuously adjusts the corresponding parameters, the SCPO algorithm can find a set of strategies that satisfy the constraints. In each training cycle, SCPO uses an optimizer to update the policy parameters and penalty terms to ensure that the constraints are met. In addition, the SCPO algorithm also uses an experience replay-based method to cache previous experience samples and randomly select sample sets to further improve training efficiency and stability. At the same time, based on the runtime verification results, constraints are used to divide and constrain the task and domain space of the agent.

After the above three stages, this framework obtains an optimized safety policy for implementing multi-agent tasks. Compared with existing work, our work can obtain a relatively optimized policy under the premise of formally guaranteeing safety. The detailed process for achieving this is described in the following subsections.

4.1. Modeling @ Runtime

We model the MRS @ runtime as a Markov Game with the extension of safety-constraint, named as Safety-Constrained Markov Game (SCMG): $< N, S, A, p, \rho, \gamma, R, C >$. Here, N is the set of agents, S is the state space, $A = \Pi_{i=1}^{n} A^i$ is the product of the agent action spaces, called the joint action space, $p : S \times A \times S \to R$ is the probability transition function, ρ is the initial state distribution, $\gamma \in [0, 1)$ is the discount factor, and R is the joint reward function. $C = \left\{ C_j^i \right\}_{1 \le j \le m}^{i \in N}$ is a set of cost functions in the form of $C_j^i : S \times A^i \to R$. At the time step t, an agent i in the state s_t takes action a_t^i according to its policy $\pi^i(a^i|s_t)$. Together with the actions of other agents, a joint action is denoted as a joint policy is $\pi(a|s) = \prod_{i=1}^{n} \pi^i(a^i|s)$. The agent receives the reward $R(s_t, a_t)$, and at the same time, each agent i pays the cost $C_j^i(s_t, a_t^i), \forall j = 1, \ldots, m^i$. Then, the multi-robot system transitions to the new state s_{t+1} with probability $p(\cdot|s_t, a_t)$. The goal of this paper is to maximize the expected reward:

$$\max_{\pi} J(\pi) \triangleq E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)] \tag{1}$$

At the same time, each agent tries to satisfy the safety constraints, written as

$$J_j^i(\pi) \triangleq E[\sum_{t=0}^{\infty} \gamma^t C_j^i(s_t, a_t^i)] \le c_j^i, \forall j = 1, \ldots, m^i \tag{2}$$

We define the policy value and state value function according to the reward as

$$Q_\pi(s, a) \triangleq E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a] \tag{3}$$

$$V_\pi(s) \triangleq E[Q_\pi(s, a)] \tag{4}$$

The joint policy π that satisfies inequality (2) is said to be feasible. It is worth noting that in the above formula, although the action of an agent i does not directly affect the cost $\left\{ C_j^k(s_t, a_t^k) \right\}_{j=1}^{m^k}$ of other agents, the action a_t^i will implicitly affect their total cost due to their dependence on the next state s_{t+1}. We use this to truly describe real-world multi-agent interactions: an agent's actions only have an immediate impact on the system locally, but other agents may be affected by its consequences at a later stage.

We denote any subset $\{i_1, \ldots, i_h\}$ of an agent by referring to its complement. Given a subset of agents $i_{1:h}$, for disjoint sets $j_{1:k}$, the multi-agent advantage function is defined as follows:

$$A_\pi^{i_{1:h}}(s, a^{j_{1:k}}, a^{i_{1:h}}) \triangleq Q_\pi^{j_{1:k}, i_{1:h}}(s, a^{j_{1:k}}, a^{i_{1:h}}) - Q_\pi^{j_{1:k}}(s, a^{j_{1:k}}) \tag{5}$$

One fact about the above multi-agent advantage function is that any advantage $A_\pi^{i_{1:h}}$ can be written as the sum of multi-agent advantages sequentially expanded by a single agent, which is calculated using GAE [7].

4.2. Runtime Verification

For SCMG, the safety property specification is expressed using probabilistic alternating-time temporal logic with rewards (rPATL), which combines PATL with expected rewards in an unknown multi-robot environment. PATL is a probabilistic extension of ATL, which is used for reasoning in multi-robot systems. For each robot in a coalition or competition, rPATL can specify a policy that ensures the probability of a safe

event occurring or that the expected reward meets a certain threshold. All formulas of rPATL refer to [41].

The syntax of rPATL is given by the following:

$$\phi ::= T \left| a \right| \neg\phi \left| \phi \wedge \phi \right| \mathrm{CP}_{\triangleright q}[\psi] \left| \mathrm{CR}^{r}_{\triangleright x}[F^{*}\phi] \right. \tag{6}$$

$$\psi ::= X\phi \left| \phi U^{\leq k} \phi \right| \phi U \phi \tag{7}$$

where $a \in AP$, $C \subseteq \Pi$, $\triangleright \in \{<, \leq, \geq, >\}$, $q \in Q \cap [0,1]$, r is the reward structure, $* \in \{0, \infty, c\}$, $k \in N$.

rPATL is a CTL-style branch-time sequential logic, in which a distinction is made between state formulas (ϕ) and path formulas (ψ). Combining with the probability operator $\mathrm{P}_{\triangleright q}$, path formula in PCTL, and the generalization of reward operator $\mathrm{R}^{r}_{\triangleright x}$, rPATL adopts the joint operator C of. ATL to specify the safety property for a multi-robot system.

There are three standard temporal operators in rPATL, namely X (next state), U (until), and $U^{\leq k}$ (qualified until).

rPATL is an enhanced CTL-style branching time sequential logic, specifically designed for analyzing and verifying systems with probabilistic and reward mechanisms. It introduces the concepts of probability and reward based on traditional ATL, allowing for more complex quantitative analysis of strategies in the system. rPATL creatively introduces a reward calculation mechanism based on PATL. Through the reward calculation mechanism, it is possible to calculate how much reward value can be accumulated when the system reaches a certain state. Ideally, the corresponding reward value can be effectively calculated, but when the target state cannot be reached, it may not be possible to give a correct reward value, so rPATL stipulates a reward type.

The basic semantics of rPATL are as follows: In rPATL, $s \models \phi$ is used to represent the state s satisfying the formula ϕ, and $\{s \in S | s \models \phi\}$ is used to represent the formula ϕ being satisfied on the state set S. Therefore, the satisfaction relation of rPATL is defined as follows:

(1) $s \models T$, always holds;
(2) $s \models a$, if and only if $a \in X(s)$;
(3) $s \models \neg\phi$, if and only if s does not satisfy ϕ;
(4) $s \models \phi 1 \wedge \phi 2$, if and only if $s \models \phi 1$ and $s \models \phi 2$;
(5) $s \models \phi 1 \vee \phi 2$, if and only if $s \models \phi 1$ or $s \models \phi 2$.

Runtime verification with PRISM-games ensures that the model SCMG satisfies safety constraints, i.e., only safe actions can be taken in the multi-robot system under consideration. As the model checker of a runtime model, PRISM-games determines whether the runtime model SCMG satisfies all of the safety constraints and functional requirements. If it does not, then the counterexample of model checking is used to refine the model operations until it satisfies safety constraints and functional requirements. Specifically, runtime verification of SCMG can follow the following steps, as shown in Figure 3:

- Model checking: Use PRISM-games modeling language to describe the runtime model SCMG, and use rPATL to specify the safety constraints and functional requirements and verify whether SCMG meets rPATL specifications.
- Counterexample discovery: If an error is discovered during model checking, then a counterexample is discovered, which is a trajectory (state and action sequence) that violates the safety property or functional requirements. These counterexamples provide information about possible unexpected behaviors of robots in the multi-robot system.
- Counterexamples-guided runtime model refinement: Based on counterexamples, the accuracy and precision of the model can be improved by dividing the states (or actions) into the refined states (or actions) or eliminating the unsafe or unexpected states (actions).

- Repeat loop: Continuously iterate model checking, counterexample discovery, and the refinement process until all of the functional requirements and safety constraints are satisfied or the refinement cannot be executed.

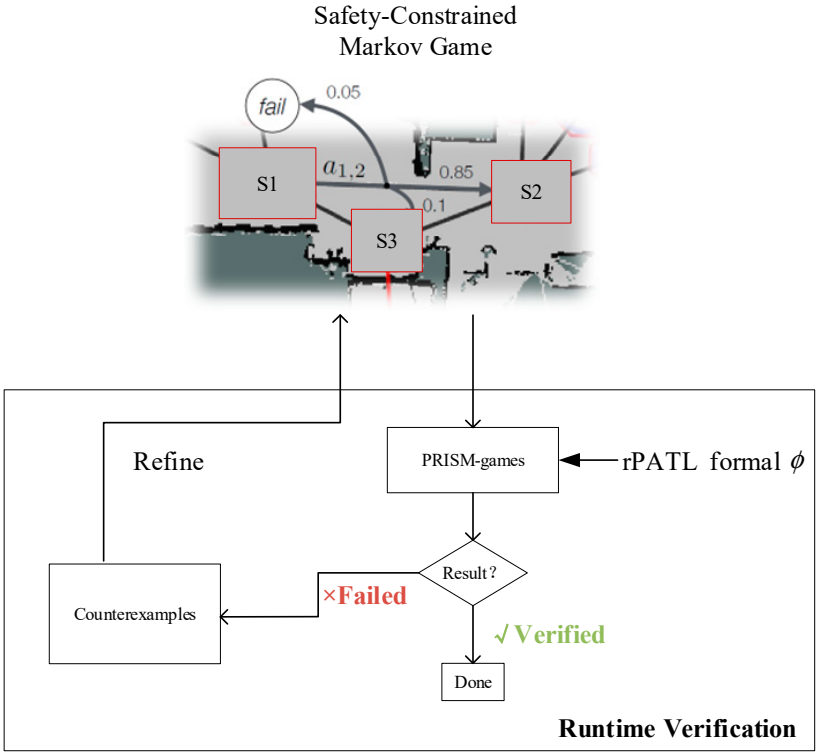

Figure 3. Runtime verification for SCMG.

Through the above steps, a multi-robot system can be verified at runtime and a verified SCMG can be provided for constraint training in the next stage. However, if the SCMG cannot satisfy the functional or safety requirements, and the refinement cannot be executed, then this may be caused by safety constraints that are too strict, or by a function that cannot be implemented. It is necessary to go back to the first stage to modify the specification regarding functional and safety constraints. Due to the cyclic and iterative nature of runtime verification, this is a continuous and time-consuming process, and this type of model checking is computationally intensive and so takes a lot of time, with the length of time depending on the scale and complexity of the system. However, runtime verification ensures that the agent always adheres to predefined safety properties, thereby reducing risk and reducing the need for adjustments or corrections after the fact. In this way, runtime verification simplifies the learning process and improves the efficiency of agent task execution, so it does not significantly affect the speed of MARL. Through the process in Figure 3, the parameters of the SCPO strategy in the next stage are adjusted according to the verified SCMG, and finally a set of strategies that meet the constraints are found. The detailed process is shown in Section 4.3.

4.3. Constraint Training

The verified runtime model SCMG can guide a multi-agent reinforcement learning algorithm to obtain safe policies. SCMG is suitable for any general MARL algorithm. Compared with existing safe MARL algorithms, the SCPO (safety-constrained policy optimization) algorithm aims to achieve collaboration between multiple agents and can maintain fast and stable learning performance when facing complex constraints. The core idea of SCPO is to use constrained optimization methods to minimize the objective function and satisfy the constraints. Specifically, SCPO uses a technique called Lagrangian Relaxation [42] to solve constrained optimization problems. It converts safety constraints into

penalties and uses a set of parameters to control the strength of penalties. As a verified runtime model SCMG continuously adjusts corresponding parameters, SCPO can find a set of policies that satisfy the constraints.

In SCPO, the agent's policy is optimized through alternate training. In each training cycle, SCPO uses an optimizer to update the policy parameters and a penalty term to ensure that constraints are satisfied. In addition, SCPO also uses an experience replay-based method to cache previous experience samples and randomly selects sample sets to further improve training efficiency and stability. When SCPO uses experience replay, the agent's behavior policy collects data through interaction with the environment, and this data (including state, action, reward, and next state) is saved in the replay buffer instead of being used to update the policy immediately. This approach enables SCPO to learn from past experience rather than relying solely on the latest data, which helps to reduce correlation in the data and ensures the safety and stability of policy updates, thereby improving overall training efficiency. Meanwhile, based on the runtime verification results, constraints are used to divide and constrain the agent's behavior, enabling the robot to complete the task without violating safety properties.

The SCPO algorithm allows joint actions so that all agents are synchronized. Synchronous action usually means that all agents take actions and update policies at the same point in time. However, this does not mean that while one agent performs its action other agents have to wait. Instead, a synchronized environment means that the environment advances to the next time step only after all agents have completed their actions for the current time step, so that each agent operates at the same time step. Therefore, a synchronized environment does not mean that agents need to wait for each other, but that the environment steps are synchronized for all agents. In the SCPO algorithm, even if an agent completes an action, then it can still help other agents that have not yet completed the action to learn by sharing information without waiting for each other.

There is a problem in that large state and action spaces prevent the agent from specifying a policy $\pi^i(\cdot|s)$ for each state individually. To solve this problem, we parameterize each agent by θ^i. Correspondingly, the joint policy π_θ is parameterized as $\theta = (\theta^1, \ldots, \theta^n)$. In each iteration, each agent i_h maximizes its agent reward and is subject to agent cost constraints. However, directly computing the max $-$ KL constraint is tricky in practical settings because it requires computing the KL divergence for each state [7]. Instead, we can relax it by taking the form of an expected KL constraint $D_{KL}(\pi_K^{i_h}, \pi^{i_h}) \leq \delta$. This expectation can be approximated by random sampling. The SCPO process may produce an infeasible policy $\pi_{\theta_{k+1}^i}$, so a feasible policy needs to be recovered by applying the TRPO [7] step to the cost alternatives, which is written as follows:

$$\theta_{k+1}^{i_h} = \theta_k^{i_h} - \alpha^j \sqrt{\frac{2\delta}{b^{i_h T}(H^{i_h})^{-1} b^{i_h}}} (H^{i_h})^{-1} b^{i_h} \tag{8}$$

where α^j is adjusted by backtracking line search, $H^{i_h} = \nabla^2_{\theta^{i_h}} D_{kl}(\pi^{i_h}_{\theta_k^{i_h}}, \pi^{i_h})\big|_{\theta^{i_h}=\theta_k^{i_h}}$ is the Hessian matrix of the average KL divergence [7] of the agent i_h, and $b_j^{i_h}$ is the agent gradient of the agent i_h. The SCPO algorithm process is shown in Algorithm 1.

The obtained policies from Algorithm 1 may also have some risks, but we can guarantee that the cumulative risk and probability of risk events are bounded based on runtime verification. It should be noted that the SCPO algorithm does not aim to obtain the global optimal policies but instead ensures that the safety constraints are embodied in the policies of all agents. The safety policies obtained from the SCPO algorithm are optimized on the current runtime model SCMG. From the point of view of the current SCMG, the safety policies obtained from the SCPO algorithm are optimal.

Algorithm 1. SCPO

1: **Initialization:** policy space $\pi^i_{\theta^i}$ with parameters θ^i, Lagrange multiplier λ^{i_h}
2: **for** iteration N = 0, 1, …, n **do**
3: A set of trajectories is collected by running the joint policy π_θ.
4: **for** each agent i **do**
5: Collect transition samples $< N, S, A, p, \rho^0, \gamma, R, C >$ using policy$\pi^i_{\theta^i}$, considering partial observability, and Share observations with other agents
6: Store transitions in the experience replay buffer
7: Sample a batch from the experience replay buffer
8: Compute the advantage function $A^{i_{1:h}}_{\pi}$ with GAE
9: Update policy $\pi^i_{\theta^i}$ by maximizing the agent's return subject to the cost constraint using an optimizer, with trust region to ensure small updates
10: Adjust λ^{i_h} using a backtracking line search to satisfy the expected cost constraint
11: Update policy $\pi^i_{\theta^i}$ by maximizing the augmented objective function $J^i_j(\pi)$, which includes intrinsic rewards and safety terms, while satisfying safety constraints
12: If the policy is infeasible, apply a TRPO step to the cost surrogate term to recover a feasible policy
13: **end for**
14: Runtime verification ensures that the policy satisfies the rPATL property $\pi_\theta \models \phi$, ensuring that the cumulative risk and the probability of risk events are bounded
15: **end for**

5. Experimental Evaluation

5.1. Case Study

In this section, we evaluate whether our framework for multi-robot systems can guarantee safety and help improve overall return. We use the multi-robot warehouse program RWARE [40] to simulate a multi-robot system of the Shanghai Intelligent Warehouse, and use python and PRISM-games to implement SCPO. The multi-robot warehouse (RWARE) environment has a grid-world warehouse where robots move and deliver the required goods, as shown in Figure 4. RWARE is inspired by real-world applications where robots pick up shelves and deliver them to workstations. The environment requires the robot (the circle) to move the requested shelf (the colored square) to the goal post (the letter "G") and back to the empty position. RWARE is a notoriously difficult environment to explore, and independent robots have been shown to struggle in this environment. A significant task for the robot in this environment is to deliver the requested shelf while also finding an empty location to return the previously delivered shelf to.

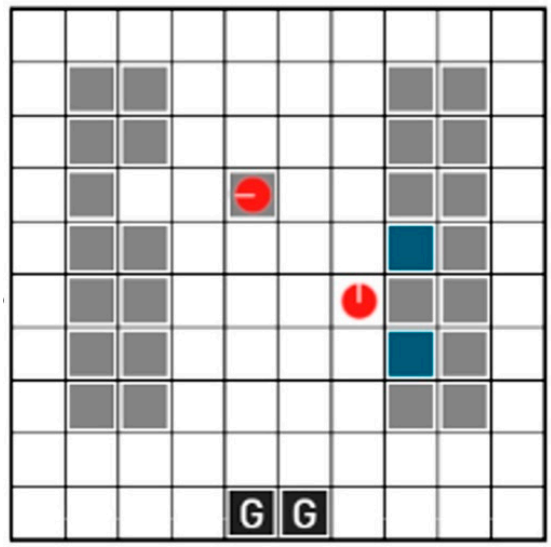

Figure 4. RWARE.

We present the experimental environment as a small (10 × 11) warehouse. We use two robots as an example, each with a control module and a transformation of disjoint subsets of action labels. The robots pick up the shelves and carry them to the workstation. Two trained agents are controlling the two robots, respectively. Here the agent has the following discrete action space: A = {move up, move down, move left, move right, load, unload}. Loading/unloading is only possible when the agent is effectively under a shelf at a pre-specified location, and the set number of shelves is two, which are randomly requested each time. In the runtime model, the agent is assumed to move between different locations without complex travel and movement within the room. This formal model building is necessary for the runtime validation of analytical models, as large and complex models cannot be analyzed using traditional computing resources. Below, we give some examples of the built SCMG.

Target. We set the goal for a robot to deliver the shelf to reach destination G, where shelf = 0 when the robot does not load the shelf, shelf = 2 when it delivers the shelf, and shelf = 1 when it returns the shelf:

label "reach_target" = (r1_x = 0 & r1_y = 4 & r1_shelf = 2) | (r2_x = 0 & r2_y = 5 & r2_shelf = 2)|(r1_x = 0 & r1_y = 5 & r1_shelf = 2) | (r2_x = 0 & r2_y = 4 & r2_shelf = 2);//Robot reaches target

Action. The action space in a multi-robot warehouse environment is very similar to a level-based foraging domain, with six discrete actions to choose from, corresponding to move up, move down, move left, move right, load shelves, and unload shelves:

player player1
robot1, [up1], [down1], [left1], [right1], [pickup1], [putdown1]
endplayer

Robot status. Each agent controls a robot designed to collect the required shelves. At any given time, two shelves are requested, and at each time step the request is delivered to the target location, and new (currently not requested) shelves are uniformly sampled and added to the request list. The agent observes a 3 × 3 grid, including information about potential close agents (given by their positions and rotations), as well as information and request lists of surrounding shelves. The action space is discrete and contains six actions, corresponding to moving up, moving down, moving left, moving right, loading shelves, and unloading shelves. The agent is rewarded only if it delivers the requested shelf to the target location.

module robot1
r1_x: [0...10] init r1_init_x;//x coordinate
r1_y: [0...9] init r1_init_y;//y coordinate
r1_shelf: [0...max_shelf] init 0;//Current shelves
//Actions and transitions
[up1] r1_y < 9 &! (r1_x = r2_x & r1_y + 1 = r2_y) -> (r1_y′ = r1_y + 1);
[down1] r1_y > 0 &! (r1_x = r2_x & r1_y − 1 = r2_y) -> (r1_y′ = r1_y − 1);
[left1] r1_x > 0 &! (r1_x − 1 = r2_x & r1_y = r2_y) -> (r1_x′ = r1_x − 1);
[right1] r1_x < 10 &! (r1_x + 1 = r2_x & r1_y = r2_y) -> (r1_x′ = r1_x + 1);
[pickup1] r1_shelf < max_shelf -> (r1_shelf′ = r1_shelf + 1);
[putdown1] r1_shelf > 0 -> (r1_shelf′ = r1_shelf − 1);
endmodule

Collision. To prevent collisions, two robots cannot be in the same position, and their movement is solved in a way that maximizes mobility. When two robots try to move to the same location, we prioritize the moving robot which also blocks the other robots. Otherwise, the selection is arbitrary. A certain selection of two shelves are requested each time. The robot cannot collect newly requested shelves without unloading previously delivered shelves.

formula collision = (r1_x = r2_x) & (r1_y = r2_y);
label "safe" = ! collision;//Safety condition
rewards "safety_violations"

collision: 1;
endrewards
Property. We use rPATL to define three properties to verify the runtime model.

Property 1. *Among all possible policies, the system has the maximum probability of completing the goal:*

<<player1, player2>>Pmax = ? [F "reach_target"] (9)

Property 2. *Among all possible policies, the minimum expected number of collisions:*

<<player1, player2>>Rmin = ? [C] (10)

Property 3. *Among all possible policies, the minimum probability that the system will eventually reach a safe state (i.e., without robot collisions):*

<<player1, player2>>Pmin = ? [F "safe"] (11)

5.2. Experimental Results

PRISM-games use a set of memory elements to resolve the choices in each state, with each memory element representing a possible "state". The memory element is updated (possibly randomly) on each transition, and the actions chosen by a robot are determined by the current memory element and the current state. If there is only one memory element, then it is memoryless; if there are a finite number, then it has limited memory. Updating memory elements and selecting actions is deterministic if it is not probabilistic; otherwise, it is updated randomly. After multiple cycles of verification and refinement in Figure 3, we obtained the safety property verification results of SCMG, as shown in Table 1.

Table 1. Property verification results.

Property	Method	Result
Property 1: <<player1, player2>>Pmax = ? [F "reach_target"]	Verification	1.0
Property 2: <<player1, player2>>Rmin = ? [C]	Verification	0.0
Property 3: <<player1, player2>>Pmin = ? [F "safe"]	Verification	1.0

The above verification results show that the maximum probability of completing the goal in the SCMG is 1, indicating that the corresponding multi-robot system can complete the task. The minimum expected number of collisions between two robots is 0, indicating that the policies of two robots can be expected to be safe and collision-free. The minimum probability in the SCMG that the robots finally reach a safe state (i.e., no robot collision) is 1, indicating that the final model we obtained through runtime verification is safe. The above results demonstrate that our framework meets the required target properties and safety properties.

Next, we use the verified SCMG to guide constraint training. First, we generate a random environment Env of size N in RWARE. We then train the SCPO agents from scratch on Env, until they converge and repeat the process for n iterations. Finally, we collect the results of these runs and start training again in the RWARE environment. It is worth noting that the SCPO algorithm achieves a safe-line search through hard constraints and backtracking.

Figure 5 shows the status of the two robots in 100 training steps during constraint training, where (r1_x, r1_y, r1_shelf) represents the (abscissa position, ordinate position, cargo status) of robot 1, and (r2_x, r2_y, r2_shelf) represents robot 2's status. If r1_x = r2_x and r1_y = r2_y, then it means that two robots collide. Otherwise, it means that two robots do not overlap in position, which indicates that there is no violation of properties 2 and 3 within 100 steps, i.e., two robots are safe and collision-free. It is demonstrated that our framework ensures the safety of the robot system before training to avoid losses during the

training process. We point out that the case we tested is computationally challenging and is characterized by the choice to access a combination of location strategies. Furthermore, our framework does not incur significant overhead compared to simpler approaches that only maximize task-satisfaction probability.

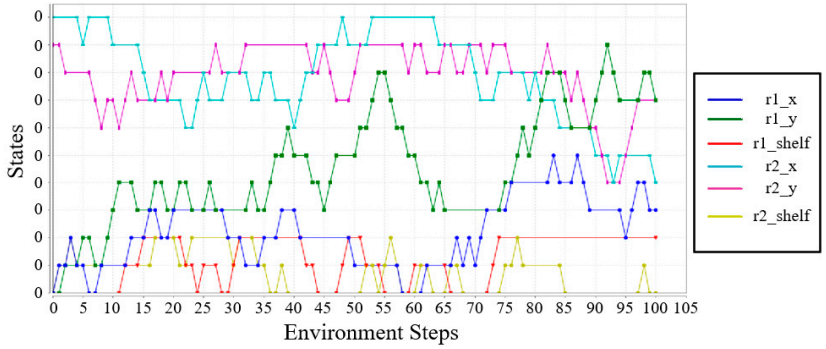

Figure 5. Constraint training status.

Figure 6 is the average training return obtained by our framework and the baseline (SNAC [38], IAC [39], and SEAC [40]) in 80 million steps of training. The mean training returns are the averages of return values obtained during the training process based on the current policy and environmental dynamics, which are used to evaluate the quality of a policy. A higher mean training return value means that the policy performs better at completing tasks in the environment. As this metric is calculated based on all evaluations performed during training, it also takes into account the learning speed in addition to the final realized return. During the constraint training process, choosing an appropriate initial policy, parameters such as constraint limits and learning rates, and providing high-quality training samples are crucial to maximizing the mean training returns. If the goal requires multiple steps to reach, then it may face time constraints, which will affect the implementation of preventive measures and lead to a decrease in safety probability. Conversely, a more conservative policy may be necessary to improve safety, but this may result in performance degradation. Therefore, when weighing safety and performance, it needs to consider different factors and find a balance point. The average episode costs of SCPO, SEAC, SNAC, and IAC are shown in Table 2. The average episode cost reflects the indicators of safety violations of different algorithms. The lower the cost value, the better the algorithm performs in satisfying security constraints. Compared with the baseline, the results show that our framework is the safest. SCPO performs worse than the state-of-the-art method SEAC during training, which is mainly because the agent's policy in our framework is subject to certain safety constraints during training. These constraints may limit the action space of the agent, thus affecting its performance. However, our framework still achieves a higher average training return than SNAC while ensuring the safety of the agent's behavior. As the number of training steps increases, the effect is similar to that of IAC. The performance of SCPO usually requires sufficient training time and iterations to gradually optimize the policy. If the training time or iterations are insufficient, then optimal performance may not be achieved, resulting in low mean training returns. In general, robots in RWARE can achieve considerable results under constraint training, that is, they can obtain an optimized safety policy. Compared with existing work based on formal verification, our framework not only focuses on the safe behaviors of agents but also achieves a certain degree of optimization of safety policy.

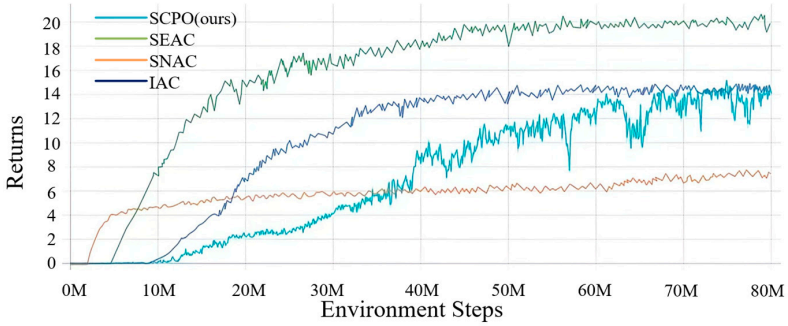

Figure 6. Mean training returns.

Table 2. Average episode cost.

Millions of Steps	SCPO	SEAC	SNAC	IAC
10 M	3	15	18	23
20 M	2	13	17	21
30 M	1	12	16	20
40 M	1	11	15	19
50 M	0	10	15	18
60 M	0	9	14	17
70 M	0	8	13	16
80 M	0	7	12	15

Figure 7 shows the optimized safety policies obtained by our framework. Module/[action] represents [robot 1 action, robot 2 action]. Each robot has six discrete actions corresponding to move up, move down, move left, move right, load the shelf, and unload the shelf. This can be seen in the actions taken by two robots to complete the goal. The results show that our framework allows robot 1 to safely and effectively achieve the goal once in step 14. That is to say, our framework can successfully implement a safe MARL in multi-robot systems. In conclusion, our framework ensures the safety of robots' behaviors before training and can obtain optimized policies in the MARL training process.

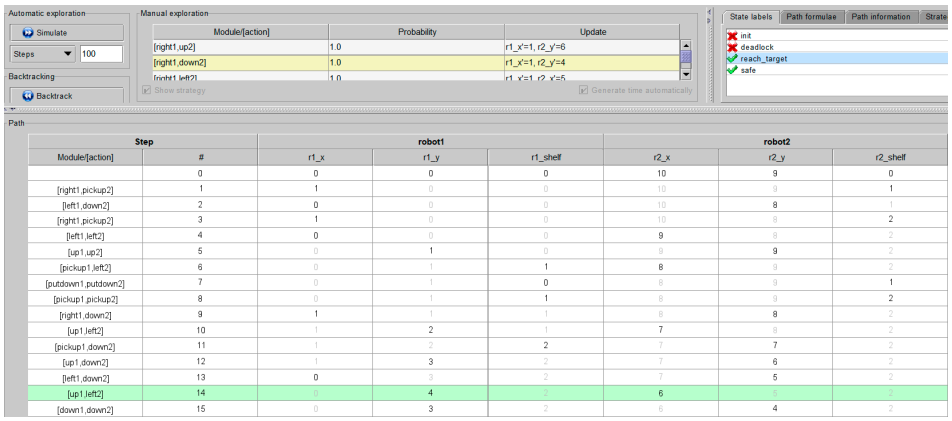

Figure 7. Optimizing safety policy simulation example.

6. Conclusions

Creating a safe MARL-based multi-robot system is an important direction for the future development of the logistics industry. It will promote digitalization, in addition to intelligent and sustainable development of the logistics industry, and will improve the efficiency and automation of various warehouse tasks. In this paper, we propose an optimized safety policy-generation framework, i.e., a safe MARL framework, for multi-robot systems in intelligent warehouses. In this framework, we extend Markov Game as SCMG to model

a multi-robot system, use rPATL runtime verification to ensure the safety of robot behaviors, and apply the verified SCMG to guide SCPO for multi-agent reinforcement learning. Finally, this framework is used in the multi-robot warehouse (RWARE) of Shanghai Intelligent Warehouse, and exhibits good performance and results, compared with existing works. It does not aim to obtain the optimal policies in terms of maximizing returns; it aims to develop an optimized safety policy that implements warehouse tasks. In the future, we will develop an automated learning runtime model method for the multi-robot system at runtime, which can improve the efficiency and correctness of modeling multi-robot systems at runtime. Moreover, designing a heuristic algorithm for counterexample generation is also an important topic for refining the runtime model SCMG. As building the runtime model accurately is difficult, this research direction is extremely valuable. Another valuable research direction is to extend this framework with multi-agent deep reinforcement learning, which may further improve the efficiency of multi-robot systems in intelligent warehouses.

Author Contributions: Conceptualization, Y.L. and J.L.; methodology, Y.L. and J.L.; software, J.L.; validation, Y.L. and J.L.; formal analysis, Y.L.; investigation, Y.L.; resources, J.L.; data curation, J.L.; writing—original draft preparation, Y.L. and J.L.; writing—review and editing, Y.L.; visualization, J.L.; supervision, Y.L.; project administration, Y.L.; funding acquisition, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the MOE Humanities and Social Sciences Foundation of China under Grant No. 20YJCZH102 and the Singapore–UK Cyber Security of EPSRC under Grant No. EP/N020170/1.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Table A1. Abbreviations and their meanings.

Abbreviations	Meanings
MRS	multi-robot systems
RL	reinforcement learning
MARL	multi-agent reinforcement learning
SCMG	safety-constrained Markov Game
RV	runtime verification
rPATL	probabilistic alternating-time temporal logic with rewards
SCPO	safety-constrained policy optimization
AGVs	automated guided vehicles
RWARE	the multi-robot warehouse
CMIX	Cooperative Multi-agent Information eXchang
QMIX	Monotonic Value Function Factorization for Deep Multi-Agent Reinforcement Learning
CMDP	Constrained Markov Decision Process
MDP	Markov Decision Process
LTL	Linear Temporal Logic
QV	quantitative verification
IPPO	independent proximal policy optimization
MAPPO	Multi-Agent Proximal Policy Optimization
HAPPO	Heterogeneous-Agent Proximal Policy Optimization
SNAC	Shared Network Actor-Critic
SEAC	Shared Experience Actor-Critic
IAC	Independent Actor-Critic
MG	Markov Game
MBSE	Model-Based Systems Engineering
GAE	Generalized Advantage Estimation
PATL	Probabilistic Alternating-time Temporal Logic
ATL	Alternating-time Temporal Logic
CTL	Computation Tree Logic
TRPO	Trust Region Policy Optimization
KL	Kullback-Leibler Divergence

References

1. Li, Z.; Barenji, A.V.; Jiang, J.; Zhong, R.Y.; Xu, G.J. A mechanism for scheduling multi robot intelligent warehouse system face with dynamic demand. *J. Intell. Manuf.* **2020**, *31*, 469–480. [CrossRef]
2. Bolu, A.; Korçak, Ö. Adaptive task planning for multi-robot smart warehouse. *IEEE Access* **2021**, *9*, 27346–27358. [CrossRef]
3. Street, C.; Pütz, S.; Mühlig, M.; Hawes, N.; Lacerda, B. Congestion-aware policy synthesis for multirobot systems. *IEEE Trans. Robot.* **2021**, *38*, 262–280. [CrossRef]
4. Hu, H.; Yang, X.; Xiao, S.; Wang, F. Anti-Conflict AGV Path Planning in Automated Container Terminals Based on Multi-Agent Reinforcement Learning. *Int. J. Prod. Res.* **2023**, *61*, 65–80. [CrossRef]
5. Sharkawy, A.N.; Koustoumpardis, P.N. Human–robot interaction: A review and analysis on variable admittance control, safety, and perspectives. *Machines* **2022**, *10*, 591. [CrossRef]
6. Choi, H.B.; Kim, J.B.; Han, Y.H.; Oh, S.W.; Kim, K. MARL-based cooperative multi-AGV control in warehouse systems. *IEEE Access* **2022**, *10*, 100478–100488. [CrossRef]
7. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep reinforcement learning: A brief survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
8. Li, Y.; Wang, X.; Sun, J.; Wang, G.; Chen, J. Self-triggered Consensus Control of Multi-agent Systems from Data. *IEEE Trans. Autom. Control* **2024**, 1–8. [CrossRef]
9. ElSayed-Aly, I.; Bharadwaj, S.; Amato, C.; Ehlers, R.; Topcu, U.; Feng, L. Safe Multi-Agent Reinforcement Learning via Shielding. In Proceedings of the 20th International Conference on Autonomous Agents and Multi Agent Systems, virtual, 3–7 May 2021.
10. Kirca, Y.S.; Degirmenci, E.; Demirci, Z.; Yazici, A.; Ozkan, M.; Ergun, S.; Kanak, A. Runtime Verification for Anomaly Detection of Robotic Systems Security. *Machines* **2023**, *11*, 166. [CrossRef]
11. Garcıa, J.; Fernández, F. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **2015**, *16*, 1437–1480.
12. Gu, S.; Grudzien Kuba, J.; Chen, Y.; Du, Y.; Yang, L.; Knoll, A.; Yang, Y. Safe Multi-Agent Reinforcement Learning for Multi-Robot Control. *Artif. Intell.* **2023**, *319*, 103905. [CrossRef]
13. Wongpiromsarn, T.; Topcu, U.; Murray, R.M. Receding Horizon Temporal Logic Planning. *IEEE Trans. Autom. Control* **2012**, *57*, 2817–2830. [CrossRef]
14. Valiente, R.; Toghi, B.; Pedarsani, R.; Fallah, Y.P. Robustness and adaptability of reinforcement learning-based cooperative autonomous driving in mixed-autonomy traffic. *IEEE Open J. Intell. Transp. Syst.* **2022**, *3*, 397–410. [CrossRef]
15. Qin, Z.; Zhang, K.; Chen, Y.; Chen, J.; Fan, C. Learning Safe Multi-Agent Control with Decentralized Neural Barrier Certificates. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
16. Hsu, K.C.; Ren, A.Z.; Nguyen, D.P.; Majumdar, A.; Fisac, J.F. Sim-to-Lab-to-Real: Safe reinforcement learning with shielding and generalization guarantees. *Artif. Intell.* **2023**, *314*, 103811. [CrossRef]
17. Liu, C.; Geng, N.; Aggarwal, V.; Lan, T.; Yang, Y.; Xu, M. Cmix: Deep multi-agent reinforcement learning with peak and average constraints. In Proceedings of the Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, 13–17 September 2021; Springer International Publishing: Berlin/Heidelberg, Germany, 2021. Part I 21. pp. 157–173.
18. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *J. Mach. Learn. Res.* **2020**, *21*, 7234–7284.
19. El Mhamdi, E.M.; Guerraoui, R.; Hendrikx, H.; Maurer, A. Dynamic Safe Interruptibility for Decentralized Multi-Agent Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 129–139.
20. Altman, E. *Constrained Markov Decision Processes*; Routledge: New York, NY, USA, 2021.
21. Yu, M.; Yang, Z.; Kolar, M.; Wang, Z. Convergent Policy Optimizeation for Safe Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 3127–3139.
22. Camacho, A.; Icarte, R.T.; Klassen, T.Q.; Valenzano, R.A.; McIlraith, S.A. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In Proceedings of the IJCAI, Macao, 10–16 August 2019; Volume 19, pp. 6065–6073.
23. Hahn, E.M.; Perez, M.; Schewe, S.; Somenzi, F.; Trivedi, A.; Wojtczak, D. Omega-Regular Objectives in Model-Free Reinforcement Learning. In Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Prague, Czech Republic, 6–11 April 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 395–412.
24. Bozkurt, A.K.; Wang, Y.; Zavlanos, M.M.; Pajic, M. Control Synthesis from Linear Temporal Logic Specifications Using Model-Free Reinforcement Learning. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 10349–10355.
25. Hasanbeig, M.; Abate, A.; Kroening, D. Cautious Reinforcement Learning with Logical Constraints. In Proceedings of the 19th International Conference on Autonomous Agents and Multi-Agent Systems, Auckland, New Zealand, 9–13 May 2020; pp. 483–491.
26. Hatanaka, W.; Yamashina, R.; Matsubara, T. Reinforcement Learning of Action and Query Policies with LTL Instructions under Uncertain Event Detector. *IEEE Robot. Autom. Lett.* **2023**, *8*, 7010–7017. [CrossRef]
27. Grimm, T.; Lettnin, D.; Hübner, M. A Survey on Formal Verification Techniques for Safety-Critical Systems-on-Chip. *Electronics* **2018**, *7*, 81. [CrossRef]
28. Ulusoy, A.; Smith, S.L.; Ding, X.C.; Belta, C.; Rus, D. Optimality and Robustness in Multi-Robot Path Planning with Temporal Logic Constraints. *Int. J. Robot. Res.* **2013**, *32*, 889–911. [CrossRef]

29. Herd, B.; Miles, S.; McBurney, P.; Luck, M. Quantitative Analysis of Multiagent Systems through Statistical Model Checking. In Proceedings of the Engineering Multi-Agent Systems: Third International Workshop, EMAS 2015, Istanbul, Turkey, 5 May 2015; Revised, Selected, and Invited Papers 3. Springer: Berlin/Heidelberg, Germany, 2015; pp. 109–130.
30. Tarasyuk, A.; Pereverzeva, I.; Troubitsyna, E.; Laibinis, L. Formal Development and Quantitative Assessment of a Resilient Multi-Robotic System. In Proceedings of the Software Engineering for Resilient Systems: 5th International Workshop, SERENE 2013, Kiev, Ukraine, 3–4 October 2013; pp. 109–124.
31. Mason, G.; Calinescu, R.; Kudenko, D.; Banks, A. Assurance in Reinforcement Learning Using Quantitative Verification. *Adv. Hybrid. Intell. Methods Models Syst. Appl.* **2018**, *85*, 71–96.
32. Mason, G.R.; Calinescu, R.C.; Kudenko, D.; Banks, A. Assured Reinforcement Learning with Formally Verified Abstract Policies. In Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART), Porto, Portugal, 24–26 February 2017.
33. Riley, J.; Calinescu, R.; Paterson, C.; Kudenko, D.; Banks, A. Reinforcement Learning with Quantitative Verification for Assured Multi-Agent Policies. In Proceedings of the 13th International Conference on Agents and Artificial Intelligence, Online, 4–6 February 2021; pp. 237–245.
34. Riley, J.; Calinescu, R.; Paterson, C.; Kudenko, D.; Banks, A. Utilising Assured Multi-Agent Reinforcement Learning within Safety-Critical Scenarios. *Procedia Comput. Sci.* **2021**, *192*, 1061–1070. [CrossRef]
35. Kadoche, E.; Gourvénec, S.; Pallud, M.; Levent, T. Marlyc: Multi-agent reinforcement learning yaw control. *Renew. Energy* **2023**, *217*, 119129. [CrossRef]
36. Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of ppo in cooperative multi-agent games. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24611–24624.
37. Kuba, J.G.; Chen, R.; Wen, M.; Wen, Y.; Sun, F.; Wang, J.; Yang, Y. Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning. In Proceedings of the ICLR 2022—10th International Conference on Learning Representations, The International Conference on Learning Representations (ICLR), Virtual, 25–29 April 2022; p. 1046.
38. Ye, Z.; Chen, Y.; Jiang, X.; Song, G.; Yang, B.; Fan, S. Improving sample efficiency in multi-agent actor-critic methods. *Appl. Intell.* **2022**, *52*, 3691–3704. [CrossRef]
39. Zhang, Y.; Zhou, Y.; Lu, H.; Fujita, H. Cooperative Multi-Agent Actor–Critic Control of Traffic Network Flow Based on Edge Computing. *Future Gener. Comput. Syst.* **2021**, *123*, 128–141. [CrossRef]
40. Christianos, F.; Schäfer, L.; Albrecht, S. Shared experience actor-critic for multi-agent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 10707–10717.
41. Kwiatkowska, M.; Norman, G.; Parker, D.; Santos, G. PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time. In Proceedings of the Computer Aided Verification: 32nd International Conference, CAV 2020, Los Angeles, CA, USA, 21–24 July 2020; Springer International Publishing: Berlin/Heidelberg, Germany, 2020. Part II 32. pp. 475–487.
42. Bragin, M.A.; Luh, P.B.; Yan, J.H.; Yu, N.; Stern, G.A. Convergence of the surrogate Lagrangian relaxation method. *J. Optim. Theory Appl.* **2015**, *164*, 173–201. [CrossRef]