robotics

MDPI

Article

Safe Reinforcement Learning for Arm Manipulation with Constrained Markov Decision Process

Patrick Adjei †, Norman Tasfi †, Santiago Gomez-Rosero *,† and Miriam A. M. Capretz †

Electrical and Computer Engineering, Western University, London, ON N6A 3K7, Canada; padjei@uwo.ca (P.A.); ntasfi@uwo.ca (N.T.); mcapretz@uwo.ca (M.A.M.C.)

* Correspondence: sgomezro@uwo.ca

† These authors contributed equally to this work.

Abstract: In the world of human–robot coexistence, ensuring safe interactions is crucial. Traditional logic-based methods often lack the intuition required for robots, particularly in complex environments where these methods fail to account for all possible scenarios. Reinforcement learning has shown promise in robotics due to its superior adaptability over traditional logic. However, the exploratory nature of reinforcement learning can jeopardize safety. This paper addresses the challenges in planning trajectories for robotic arm manipulators in dynamic environments. In addition, this paper highlights the pitfalls of multiple reward compositions that are susceptible to reward hacking. A novel method with a simplified reward and constraint formulation is proposed. This enables the robot arm to avoid a nonstationary obstacle that never resets, enhancing operational safety. The proposed approach combines scalarized expected returns with a constrained Markov decision process through a Lagrange multiplier, resulting in better performance. The scalarization component uses the indicator cost function value, directly sampled from the replay buffer, as an additional scaling factor. This method is particularly effective in dynamic environments where conditions change continually, as opposed to approaches relying solely on the expected cost scaled by a Lagrange multiplier.

Keywords: safe reinforcement learning; constrained Markov decision process; Lagrangian multiplier; scalarized expected return; UR5 arm robot

Citation: Adjei, P.; Tasfi, N.; Gomez-Rosero, S.; Capretz, M.A.M. Safe Reinforcement Learning for Arm Manipulation with Constrained Markov Decision Process. *Robotics* **2024**, *13*, 63. https://doi.org/10.3390/robotics13040063

Academic Editors: Dan Zhang and Xinjun Liu

Received: 17 February 2024
Revised: 25 March 2024
Accepted: 12 April 2024
Published: 18 April 2024

1. Introduction

The world of human and robot coexistence is expanding. Without intuitive behavior from the robot, shared space between the machine and the human becomes dangerous [1]. Unfortunately, intuition cannot be constructed with traditional logic approaches because they require an explicit representation of all possible states and actions. This is not feasible in complex environments [2] and is unattainable at the design phase [3]. Reinforcement learning (RL) with neural networks, which is referred to as RL throughout this paper, has demonstrated outstanding results and can generalize actions from possible states [4,5]. However, the caveat is that most RL approaches learn policies that progressively improve the reward criteria through exploratory actions, which can be problematic [4,6,7]. This can compromise safety in the Markov decision process (MDP), where MDP represents the sequential decision process in RL, and random, non-strategic exploratory actions can inadvertently lead the system into hazardous or non-functional states [8]. To mitigate such risks, it is important to engage in strategic exploration, meaning that exploration should not be random, but rather guided by strategies that consider the safety and operational integrity of the system. Hence, integrating strategic exploration becomes imperative for balancing the dual objectives of effective learning and ensuring safety within the RL paradigm.

There is a taxonomy of safety definitions aligned with obstacle avoidance. For instance, fulfilling ergodicity is the ability to reach any other state from a given state [8], implying that the system must remain operational and adaptable across all states. Another definition of safety requires humans to label states of environments as either *safe* or *unsafe*. An agent

is considered safe if it never enters any of the uncontrolled states labeled unsafe where no policy can make a permissible recovery [9], thereby aligning its operational parameters with human-defined safety constraints. This aligns with constraining the agent from unsafe states [10]. In the RL literature, this approach is known as the constrained Markov decision process (CMDP), which incorporates these constraints directly into the decision-making framework. Garcia and Fernandez [11] have tabulated other approaches to achieving safety. Among those that involve modifying the optimization criterion, the CMDP approach is used on a single-arm manipulator to integrate safety at a fundamental level in decision-making. This study aimed to achieve motion planning with a simulated UR5 arm manipulator while avoiding collision with a nonstationary object that does not reset.

Figure 1 shows the UR5 arm, a simulated industrial robot with six rotating joints designed for automating repetitive tasks in various settings. From an initial state, the objective is for the arm to reach a goal state without colliding with a moving object, a challenge that requires precise and adaptive control strategies. Much of the work that addresses planning problems with an arm manipulator simply shapes the reward function as a composition of different scenarios. This can lead to the occurrence of *reward hacking*, which is the result of the designer misspecifying the intended objective function [12]. In complex environments, particularly when the reward function is sophisticated and may not adequately capture all nuances of the dynamics for a specific task, such a compositional approach is susceptible to *reward hacking*.

Note that a real robot raises more complications than one that is simulated. Although simulation may simplify the task, it is a safeguard against sensor and motor component burnout because of the long duration of RL training. This makes it adequate for testing RL algorithms without damaging the robot components. Therefore, the initial step towards safety in motion planning on the UR5 robot was carried out by simulation, where simulated obstacles in the real world are modeled with floating spheres.

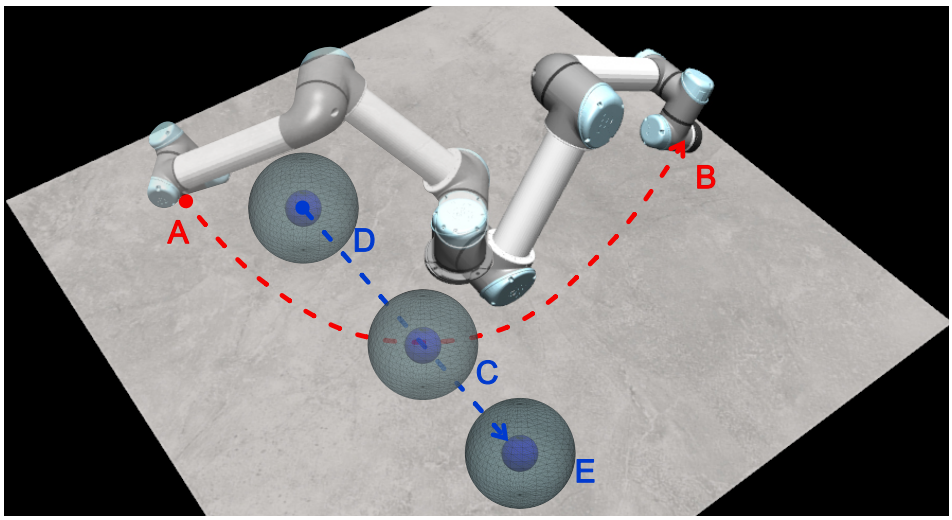

Figure 1. Configuration setup for the UR5 and the moving object. UR5 arm moving from point A to B and learning how to avoid collision with an object moving from point D to E.

The contribution of this paper is approaching arm manipulation with CMDP using the Lagrange multiplier (LM) combined with scalarized expected return (SER). The distinguishing portion of the proposed method is combining the indicator cost function values sampled from the replay buffer as the scaling component of SER. This acts as an adaptive mechanism that toggles the objective based on the label of the region. The results indicate a significant improvement over the CMDP strategy based solely on LM, particularly in dynamic environments where the object does not reset to an initial position. Moreover, this approach simplifies the reward function by reducing the number of terms, making it more

attractive and effective for addressing the *reward hacking* problem that is often encountered in complex manipulative tasks.

The sections of this paper are structured as follows. Section 2 begins with a background presentation that introduces the mathematical notations used throughout the paper, laying the foundation for subsequent discussions. This is followed by Section 3, which reviews related work, focusing on various approaches to safety and examining studies that involve shaping the reward for obstacle avoidance in robotic arm manipulators by considering multiple scenarios. Section 4 details the proposed methodology. The results, presented in Section 5, offer a detailed comparative analysis, highlighting the effectiveness of the method compared with using only LM in CMDP formulation. Finally, Section 6 concludes the paper, summarizing the findings given in the contributions and outlining possible future directions.

2. Background

This section outlines the foundational background that is needed for the proposed method. It begins with an overview of traditional MDP and progressively builds towards the concept of combining scalarization with Lagrangian methods in CMDP.

2.1. Markov Decision Process

Reinforcement learning is a machine learning paradigm that uses the MDP framework modeled by $(S, A, \mathcal{T}, r, \gamma)$ [13,14]. In the MDP, in each timestep, t, the system provides all necessary information for a decision-maker to take an action, $a_t \in A$, at time t, with A being the action space. The information provided by the system corresponds to the system's state, $s_t \in \mathcal{S}$, at time t, with S being the state space. The action, a_t, is governed by a policy, $\pi(a|s) \longrightarrow (0, 1)$. A policy governs the behavior of an agent. It specifies what action the agent should take in each state to maximize its expected reward over time. In our case, the policy is a probability distribution over the available actions in a given state. In state s_t, an action, a_t, is taken, perturbing the system to a new state, $s_{t+1} \in S$, based on the environment transition dynamics $\mathcal{T}(s_{t+1}|s, a)$. The transition dynamics is a model of the system that captures the underlying rules or mechanisms that govern how the system evolves from one state to another. After transitioning, the agent receives a reward signal, $r_t \in \mathbb{R}$, based on the reward function $r : S \times A \longrightarrow \mathbb{R}$. The reward signal dictates the quality of the action taken, which is used to update π accordingly. In the deep learning setting, the policy is a neural network; therefore, the update is made on the parameters of the neural network $\theta \in \mathbb{R}^k$. Without loss of generality, the θ parameterization is omitted from the policy in the background and made more explicit in the methodology section. Overall, the objective of the agent is to maximize the expected discounted reward by selecting the appropriate actions. The expectation is denoted as $\mathbf{E}[\cdot]$. Conceptually, the agent should learn the optimal policy

$$\pi^* = \arg\max_{\pi} \mathbf{E}_{a\sim\pi}\Big[\sum r(s, a)\Big] \tag{1}$$

that yields the most reward for every state from selecting an action.

2.2. Soft Actor-Critic

For complex learning domains that are high-dimensional and continuous in state-action spaces, such as an arm manipulator environment, it is difficult to find exact solutions for the MDP when using action-value-based algorithms [15,16]. The soft actor–critic (SAC) algorithm handles continuous state-action space problems and alters the objective function by including an entropy term [17], redefining Equation (1) as

$$\pi^* = \arg\max_{\pi} \mathbf{E}_{a\sim\pi}\Big[\sum r(s, a) + \alpha\mathcal{H}(\pi(\cdot|s))\Big]. \tag{2}$$

where $\mathcal{H}$ is the entropy function and α weights the importance of the entropy term. This extra term brings many benefits. The prominent one is encouraging exploration; therefore, the agent's aim is to maximize the reward, even while behaving randomly.

2.3. Constrained Markov Decision Process

The constrained MDP (CMDP) [10,18] is modeled by $(S, A, \mathcal{T}, r, c, d, \gamma)$, where, like the reward r, c is a random variable modeled by a cost function, $c : S \times A \longrightarrow \{0, 1\}$, and $d \in \mathbb{R}$ is a parameter for the allowable cost limit that controls the level of penalty imposed by a violation. In RL, a Q-function represents the expected cumulative reward that an agent can obtain by starting from a specific state and taking a particular action. The Q-function is denoted as $Q_r^\pi(s,a) = \mathbf{E}_{a\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)|s = s_t, a = a_t\right]$ to represent the expected return for the random variable r when an action, a, is taken in state s and following π thereafter. Like the Q-function for the reward, the Q-function for the cost, c, is represented as $Q_c^\pi(s,a) = \mathbf{E}_{a\sim\pi}[\sum_{t=0}^{\infty}\gamma^t c(s_t,a_t)|s_t = s, a_t = a]$. Approaching safety in RL with CMDP is to find the optimal policy, π^*, which simultaneously satisfies the constraining function and the cost limit

$$\pi^* = \arg\max_\pi Q_r^\pi \quad s.t. \quad Q_c^\pi \leq d. \tag{3}$$

2.4. Lagrange Multiplier

A constrained optimization problem can be formulated in the context of reinforcement learning (RL) using a Lagrange multiplier (LM) approach [19]. This methodology involves finding a policy that not only maximizes expected rewards, but also minimizes the expected cost, keeping it within a tolerable limit. In essence, solving Equation (3) as an LM optimization problem represents this dual focus on reward maximization and cost minimization within the RL framework as

$$\max_\pi \min_\lambda L(\pi,\lambda) = Q_r^\pi(s_t,a_t) - \lambda(Q_c^\pi(s_t,a_t) - d). \tag{4}$$

In this scenario, the agent is directed to optimize Q_r, the estimate of the reward function, while simultaneously being penalized for any estimate of the cost, Q_c, that violates the limit, d. $\lambda \in \mathbb{R}$ acts as a coefficient to outweigh the estimation of the reward. Depending on the limit violation, this parameter is learned throughout the learning process. This weighting mechanism ensures that, as Q_c becomes more significant relative to Q_r through the influence of λ, the agent increasingly prioritizes adherence to cost constraints, thus aligning its actions more closely with the desired safety and operational parameters designed by the user.

2.5. Scalarized Expected Return

Scalarized expected return (SER) is a key concept in multi-objective RL [20], where the agent's satisfaction is determined by multiple outcome values, typically represented in vector form as returns. In multi-objective RL, scalarization is used to manage the trade-offs between these various outcomes effectively. Specifically, within the SER framework, the process begins by calculating the expected value of these returns for a given policy. Given that Q symbolizes an expectation in this context, Equation (4) can be formulated to reflect this scalarization approach as

$$\max_\pi \min_\lambda L(\pi,\lambda) = (1-c)Q_r^\pi(s_t,a_t) - c\lambda(Q_c^\pi(s_t,a_t) - d), \tag{5}$$

where the coefficient c is the immediate cost, which defines the cost to be either 1 or 0. This acts to toggle the optimization objective depending on the label of the region.

3. Related Work

According to Garcia and Fernandez [11], solely maximizing long-term reward in reinforcement learning (RL) may overlook infrequent but significant instances of large negative outcomes. This necessitates a transformation of the optimization criterion to maximize expected return while accounting for potential violations simultaneously. Unfortunately, existing literature on RL with arm manipulators predominantly focuses on composing reward functions for collision avoidance, often neglecting the application of more nuanced safety approaches like risk-based, uncertainty-based, and CMDP-based methods. The upcoming subsections will explore various studies that address these different dimensions of safety within RL. In addition, the research specifically concerning arm manipulators using RL for obstacle avoidance is discussed, which is in accordance with safety in the robotics domain.

3.1. Risk-Based Safety in Reinforcement Learning

A popular approach is to modify the RL algorithm to account for risk in the formulation [21–23]. In this risk-sensitive approach, the optimization criterion is transformed to include a measure that reflects a trade-off between risk and reward by including a parameter that controls sensitivity in stochastic environments [11]. Jaimungal et al. [21] used the rank-dependent expected utility function in their RL formulation to trade off risk and reward. Their objective was to minimize a measure of risk, subject to model uncertainty caused by distorted states, where the distortion was produced by a distribution to have the worst performance possible within the Wasserstein ball. Geibel and Wysotzki [22] solved the problem with two value functions. They aimed to maximize the initial value function, subject to the added value function serving as a cost signal that is constrained by weight to dictate the feasibility of the policy. Mihatsch and Neuneier [23] created a risk framework by transforming the temporal difference used for learning. A variable controls the level of risk to overweight, underweight, or neutralize the successor state. When the variable is positive, it weights the negative temporal difference more heavily, making it risk-avoidant. Setting the variable to zero makes it risk-neutral, and when the variable is less than zero this represents risk-seeking. Safety in the proposed approach is ensured by constraining the agent with a discounted cost function. The risk-based approach to safety in RL incorporates safety considerations through the RL objective or reward function, whereas constraint MDP (CMDP) explicitly models safety constraints as part of the MDP formulation.

3.2. Uncertainty-Based Safety in Reinforcement Learning

Uncertainty-based safety approaches are variabilities that are either due to inherent stochasticity of the system or to stochasticity in the MDP parameters [11]. To account for uncertainty, the agent must be ready for the worst case when the model is inaccurate [24], by formulating a worst-case optimization criterion [11]. For example, to verify system properties with linear temporal logic, Wolff et al. [25] generated a control policy to maximize the worst-case probability of satisfying the linear temporal logic based on their defined specifications to account for the uncertainties in the transition dynamics. Typically, using worst-case criteria to account for uncertainty in the transition dynamics does not include safety constraints in the formulation. Russel et al. [26] addressed this by integrating CMDP into the formulation of a robust MDP to account for transition probability uncertainty in the MDP. Furthermore, Bossens and Bishop [24] developed a framework incorporating robust MDP and CMDP, but the safety constraints are enforced by an escape mechanism that monitors the system state. Near constraint budget violation, the escape mechanism switches to a safety policy for actions that aim to return the system to a known safe state. To capture environment variations, Chen et al. [27] adopted context-awareness using attentive neural processing. With the use of prior offline data acting as context, the attentive neural processing unit captures the difference between the prior and the environment samples as a disturbance error. By minimizing this error, including with their additional dual constrained objective, their goal is to improve the model prediction of future safe states and

safe actions in a non-stationary environment. The proposed method approaches safety with CMDP, where constraints are imposed throughout the MDP process, whereas uncertainty-based approach safety is achieved through stochasticity in the system or variability in the MDP parameters.

3.3. Constrained MDP-Based Safety in Reinforcement Learning

In CMDP, the objective is to maximize the expectation of return, subject to constraints on the policy [10,11]. The work performed by Wachi and Sui [28] expanded a pessimistic safe region, where the agent must satisfy a safety constraint with high probability, with the ability for the agent to return to a set of safe states. Meanwhile, their optimistic region considered a safe state, even with a small probability that the agent could return to a safe state set. Although their approach is generic, the problem presented in this paper is tackled using an approach with LM and a non-deterministic policy. Borkar and Jain [29] presented an approach for solving a finite-horizon stochastic dynamic optimization problem with a conditional value-at-risk (CVaR) constraint. Instead of a CVaR constraint in each timestep, their constraint formulation applies to the entire trajectory, simplifying the time consistency problem. The proposed formulation dives directly into LM formulation without a CVaR. Achiam et al. [18] proposed using a trust region for policy update to guarantee constraint satisfaction during training to tackle the non-guarantees of finding the optimal solutions in model-free RL. This approach is a candidate direction to achieving safety with arm manipulation; however, the proposed approach does not consist of trust regions to update the policy. Instead, the arm manipulation problem is solved using LM and the CMDP formulation combined with scalarized expected return using the cost samples directly, which acts to toggle the objective towards either optimizing to avoid cost or preference for reward.

3.4. Reinforcement Learning Obstacle Avoidance with Arm Manipulators

Obstacle avoidance has been a trending topic in RL to enforce safety. The work described in this paper focuses on obstacle avoidance for the UR5 arm manipulator. For a dual-arm space project, Li et al. [30] strictly modified the reward function to avoid collisions between arms and to avoid collision between the end-effector (EE), also known as the tip of the arm and the target obstacle. Cao et al. [31] used the same construct with dual arm manipulators; however, in their methodology, they formulated a linear combination of two policies that had prior experience to avoid the cold-start problem. Li et al. [32] increased collision awareness by appending collision observations to an additional collision buffer. This enabled the agent to learn the state and action pair that led to a collision in every learning iteration. For their reward guidance, Tang et al. [33] fused multiple reward schemes. One scheme served for collision detection to handle non-regular-shaped obstacles, and another was guided by marked points to avoid collision using a repulsive force function, which is commonly implemented by applying an increasing penalty as the robot approaches an obstacle. Similarly, Sangiovanni et al. [34] used different reward compositions, one of which was a repulsion function to avoid collision of a non-static object. Although dual-arm manipulators are more challenging in their sparse reward approach, Prianto et al. [35] penalized the agent when a collision occurred, and their obstacle was static. Zeng et al. [36] also used different reward schemes to form an overall reward function; however, their novel scheme was formulated based on a manipulability index modeled with the Jacobian. They expressed the manipulability index as the allowable control when performing the task. In addition to a collision avoidance reward scheme for the EE, Yang and Wang [37] explicitly attempted to solve for feasible pose angles of the arm robot, given the position of the EE, so that the links of the robot also avoided the obstacle. Kamali et al. [38] combined deep RL and dynamic path following in an arm robot. Using a controller as a track point, the objective function was for the EE to follow the track point while avoiding objects. Similarly to previous work, their formulation to avoid obstacles was a composition of reward functions. Simply shaping the reward

through compositions of reward functions for robotic arm obstacle avoidance remains a common approach in recent work. Avaie et al. [39] addressed static obstacle avoidance with a complex form of a reward function that considered many aspects of arm maneuverability to comply with safe motions and to adhere to human preferences. The work described in this paper does not explicitly augment the reward function for object detection or for collision avoidance. Instead, the motion planning problem is tackled using a Lagrangian objective function plus an additional scale on the reward and cost expectations. This scale is based on the cost values from the replay buffer, which are used in addition to the Lagrange cost multiplier coefficient.

4. Methodology

The objective function serves to achieve motion planning with the UR5 arm manipulator while avoiding collision during the trajectory. The safety aspect that is emphasized in this paper is robotic motion without collision. This is attainable by including obstacle coordinates relative to each nonstationary link position of the arm manipulator within the agent's state space. Note that low-level trajectory control is not part of this scope. When used in real robot scenarios, a control trajectory system handles the low-level control to alleviate the concerns about forbidden positions and avoid links colliding with each other. Therefore, without the need for such concerns, the initial step in testing the feasibility of the algorithm was to simulate the UR5 in a MuJoCo [40] environment. MuJoCo is a high-performance physics engine designed for model-based control within robotics. The MuJoCo engine accurately simulates complex physical interactions involving multiple joints and contact points. This makes it an ideal platform to test and refine rigorously designed algorithms for robotic manipulation.

4.1. State Features

In each state, the agent can access the arm robot's center of mass for each axis. The center of mass for the x-axis is computed as $m_x = \frac{\sum_i^n m_i x_i}{\sum_i^n m_i}$, where x_i is the corresponding link position on the x-axis and m is the mass of the link. The mass (*model.body_mass* function is used to retrieve the mass of individual bodies in the MuJoCo physics simulation) is acquired from the MuJoCo simulation engine. m_y and m_z are also computed for the y- and z-axes. The angular velocity (*data.qvel* function is used to retrieve the angular velocity of each joint in the MuJoCo physics simulation) of each joint is acquired from MuJoCo. Similarly, the angles (*data.qpos* function is used to retrieve the angles of each joint in radians in the MuJoCo physics simulation) of each joint in radians are provided by MuJoCo. However, the radians are normalized between $\pm 2\omega$, where $\omega \in [0, 3.1415)$. The EE position 3D coordinate is provided in each timestep, as well as the distance (*find_closest_point* PyVista function is used to find the index of the closest point in a mesh object with Euclidean distance) of each nonstationary link to the closest point on the obstacle, O. Similarly, the distance of each link to the closest point on the sphere, H, is computed. The 3D coordinates of each nonstationary link and the object, O, are provided in each timestep.

4.2. Reward Function

The reward function is formulated using the negative Euclidean distance between the EE and the goal location:

$$r = -\sqrt{(x_t - x_g)^2 + (y_t - y_g)^2 + (z_t - z_g)^2}, \quad (6)$$

where x_t, y_t, z_t are the x-, y-, and z-coordinates of the EE in each timestep, t, and x_g-, y_g-, z_g- are the x-, y-, and z-coordinates of the goal position. This results in an increased reward as the EE approaches the target coordinates.

4.3. Danger Region and Cost Function

For simplicity, an obstacle, O, is modeled as a sphere, and O_s is the space occupied by O. Another sphere is created as a layer around O called H, and H_s is the space occupied by H. Both O and H always have the same center coordinates. The cost function is an indicator function defined by c [41]. Then:

$$c = \begin{cases} 1, & \text{if } l_i \in H_s \mid\mid O_s \\ 0, & otherwise, \end{cases} \tag{7}$$

where $l_i \in L$ is the set of nonstationary link coordinates, including the EE. c takes on a value of 1 when l_i is in H_s or in O_s. However, when l_i is in O_s, the arm position resets to the initial position, but the object, O, does not reset to the initial position. The non-resetting of the object is important because it makes the problem more challenging and closer to real-world scenarios. When the obstacle and the arm are reset, the agent needs to learn one trajectory to avoid the obstacle, which becomes similar to a static object problem. Spheres O and H are created with the PyVista package.

4.4. Soft-Actor Critic Training

The implementation of the SAC algorithm consists of two critic networks. The critic on the network parameters ψ provides feedback for the reward function (6). The critic on the network parameters ϕ provides feedback on the cost defined by (7). The actor–network has parameters, θ, that define the policy. All the networks are of the same size; therefore $\phi, \psi \in \mathbb{R}^k$. Each corresponding critic has a target network represented by $\bar{\psi}$ and $\bar{\phi}$, respectively, that undergoes soft updates during the learning phase. The output of the actor–network is used as a parameter to sample from a Gaussian distribution; therefore, for $\pi_\theta(a|s), a \sim \mathcal{N}(\mu, \sigma) + e_t$, where μ and σ are the mean and variance, respectively, output from the actor–network and e_t is an input noise sample from a fixed distribution. All networks are trained at every timestep, t. Computing the loss for the critic, parameterized by ϕ,

$$J_{Q_{c,\phi}} = \mathbf{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q_{c,\phi}(s_t,a_t) - \hat{Q}_{c,\bar{\phi}}(s_t,a_t)\right)^2\right], \tag{8}$$

where

$$\hat{Q}_{c,\bar{\phi}}(s_t,a_t) = \mathbf{E}_{a\sim\pi_\theta(\cdot|s)}\left[c(s_t,a_t) + Q_{c,\bar{\phi}}(s_{t+1},a)\right]$$

D is the replay buffer from which s_t and a_t are sampled. Now, computing the loss for the critic, parameterized by ψ,

$$J_{Q_{r,\psi}} = \mathbf{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q_{r,\psi}(s_t,a_t) - \min_{i=1,2}\hat{Q}_{r,\bar{\psi}_i}(s_t,a_t)\right)^2\right], \tag{9}$$

where

$$\hat{Q}_{r,\bar{\psi}_i}(s_t,a_t) = \mathbf{E}_{a\sim\pi_\theta(\cdot|s)}\left[r(s_t,a_t) + Q_{r,\bar{\psi}_i}(s_{t+1},a) - \alpha\log\pi_\theta(a'|s_{t+1})\right].$$

The policy is trained to fulfill a dual objective that consists of maximizing the expected reward, considering the entropy of the policy, and minimizing the expected cost scaled by λ. As in Haarnoja et al. [17], a Gaussian noise is added in the action selection process to encourage exploration. Adding noise to the sampling action, given the state, let $a_t = f_\theta(e_t; s_t)$. Using SER from Equation (5), the objective function is now

$$J_{\pi_\theta} = \mathbf{E}_{(s_t,a_t)\sim\mathcal{D}}\Big\{(1-c)\Big[\alpha\log\pi_\theta(f(e_t;s_t)|s_t) - \min_{i=1,2}Q_{r,\psi_i}(s_t,f(e_t;s_t))\Big] + c\lambda Q_{c,\phi}(s_t,a_t)\Big\}. \tag{10}$$

The policy is structured to optimize a linear scalarized combination of expected return and cost, in addition to the Lagrange multiplier acting as a scaling factor on the cost component. This design effectively creates a dynamic toggling mechanism: when the agent

is not in the hazardous state, H_s, it focuses on maximizing the expected return; conversely, when it enters H_s, the emphasis shifts towards minimizing the expected cost. This adaptive strategy enables the policy to balance between achieving its goals and maintaining safety under varying conditions, even without object resetting.

4.5. Lambda Training

In the context of Lagrange optimization, λ is the multiplier, which is crucial for controlling cost. λ is updated as a function of Softplus times the proportion of expected cost:

$$J_\lambda = \sum_t \frac{1}{\beta} \log(1 + \exp(\beta\lambda_t))(Q_{c,\phi_j}(s_t, a_t) - d), \tag{11}$$

as the agent violates the constraint limit, lambda increases to put more weight on Q_c in (10). λ is updated every other step to discourage an early conservative policy.

5. Results and Discussion

This section delineates the configuration of the neural networks and parameters used in the experiments. Subsequently, it presents the results of the proposed method and provides an analysis of the cost limit. Finally, the results and limitations are discussed.

5.1. Neural Network Details

This study uses two feedforward neural networks, each featuring two hidden layers with 128 neurons, to control a UR5 robotic arm. These networks are distinct primarily in their output layers to suit their specific functions. For the critic networks, a single neuron in the output layer is designated to calculate a Q value for each state, which is crucial for assessing the actions proposed by the actor. On the other hand, the output layer of the actor–network is made out of six neurons corresponding to the six joints of the UR5 arm, with each neuron responsible for each joint angle. To enhance decision-making throughout the learning process, the Adam optimizer is used to learn the weights of the networks.

5.2. Experimental Setup

The experiments were conducted on a desktop PC equipped with an Intel Core i9-7900X processor, an NVIDIA GeForce RTX 3090 GPU, and 32 GB of DDR4 memory. The implementation was in Python (v 3.8.16), and the major packages used were PyVista (v 0.38.5) to model the object and PyTorch (v 1.13.1+cuda 11.7) for neural network training; MuJoCo (v 2.3.3) was used as the simulation environment for the UR5 arm robot. The project folder with all the related files and source code modules can be found in the pML0x0/Scalarized-LM-SAC GitHub repository.

For a consistent analysis, both methods were tested under identical hyperparameters. Table 1 details the experiment's hyperparameters and their respective values. To clarify some of these parameters, the state features, when combined, result in a state dimension of 48. This dimensionality corresponds to the inputs that the agent considers in its decision-making. The output of the actor–network, which consists of six neurons, matches the dimension of the action space, with each neuron influencing one aspect of the arm movement.

The parameter that controls exploration over SAC is α. Table 2 shows the results for the selection of α. Given the average violation counts and average collision counts over five seeds and 6000 episodes, a constant value of $\alpha = 0.05$ is selected.

The initial value of λ is set to -10 to encourage initial exploration; a higher starting point could make the agent too conservative in decision-making. The learning rate of λ determines how significantly λ is adjusted in response to the expected cost. The β parameter is critical in defining the steepness of the Softplus function, which affects how abruptly λ is updated. The soft-update parameter forms a linear combination with the local and target networks, ensuring better stability in the learning process as opposed to having the same network also act as the target. Finally, the action range, expressed in radians,

specifies the maximum rotational movement allowed for each robot joint in a single action step. A small range is chosen to encourage safe movements of the robotic arm.

Table 1. Hyperparameter values used to run the experiments.

Hyperparameter	Symbol	Value
State dimension	-	48
Action dimension	-	6
Discount factor	γ	0.99999
Initial temperature parameter	α	0.05
Cost limit	d	0.001
Initial lambda	λ	-10
Lambda learning rate	-	0.001
Softplus Beta	β	0.5
Network learning rate	-	0.0001
Soft-update parameter	-	0.005
Object (radius)	O	0.2
Sphere (radius)	H	0.35
Replay buffer	D	300,000
Batch size	-	256
Action range in radians	-	$\pm$0.0698132

Table 2. Alpha hyperparameter tuning results.

Alpha (α)	0.05	**0.1**	**0.2**	**0.3**	**0.4**
Violations	14,812.75	15,188.5	21,314.0	15,845.25	24,864.25
Collision	168.75	214.75	681.5	256.0	590.0

5.3. Performance Comparison

This subsection presents the results of the proposed approach, which combines scalarized expectation with Lagrange multipliers (LM) and is compared with the method of solely using LM in a dynamic UR5 environment where the object to be avoided does not reset. Three key metrics are used to draw a comprehensive comparison: the cumulative number of violations, the total count of collisions, and the average reward achieved per episode. In the studies presented, the figures display the various metrics on the y-axis and their progression over episodes on the x-axis. The range of the plots is represented by the minimum and maximum values across five seeds, with the solid line indicating the average.

Figure 2 shows the average reward for the proposed method compared with solely using LM. The average reward also depends on the number of timesteps taken to reach the goal. Given the negative reward function (6), the more steps that are taken in an episode, the lower will be the average reward. In the early training phase, the number of steps can be increased because the agent is more exploratory. However, the nature of exploration can also cause the agent either to collide more often or to reach the goal state more by chance. This results in higher variation in the early stage of training.

Figure 3 shows the performance of the proposed model over violations and collisions. A violation occurs when any arm link in movement, l_i, comes close to colliding with an object, but does not hit the object; then $l_i \in H_s \wedge l_i \notin O_s$. The cumulative number of violations is shown in Figure 3a. A collision occurs when any arm link in movement hits the object; then $l_i \in O_s$. Figure 3b shows the cumulative number of collisions.

Figure 4 provides an analysis of the different success rates. In concrete terms, collision-free corresponds to a successful trajectory where none of the robot's links collide with the object but may enter the violation region, hence $l_i \in H_s \wedge l_i \notin O_s$. Violation-free corresponds to a successful trajectory without considering whether a violation occurred or a collision. The method achieves rates of 84% and 98% for violation-free and collision-free, respectively.

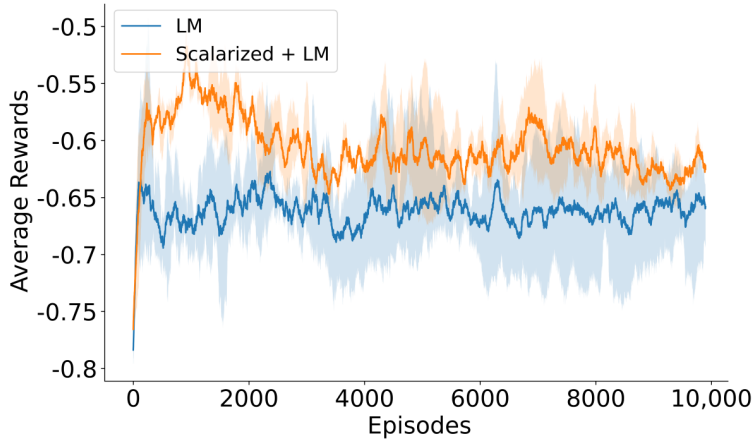

Figure 2. Averagereward progress over episodes. The blue curve represents the LM method alone, and the orange curve is the proposed scalarized + LM method.

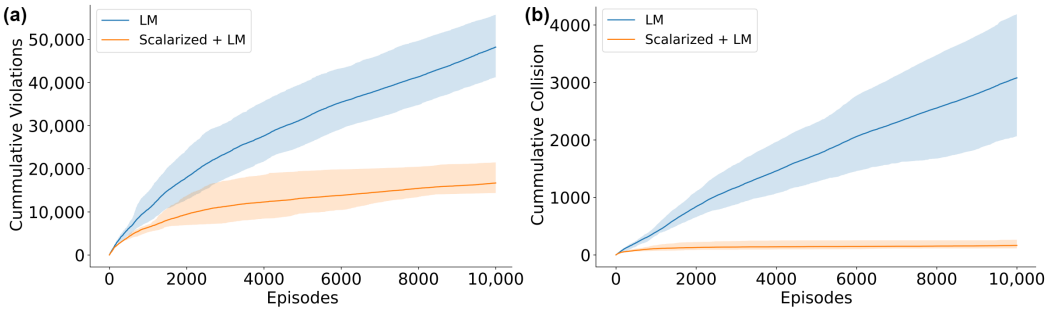

Figure 3. Performanceover violations and collisions: (**a**) cumulative violations and (**b**) cumulative collisions.

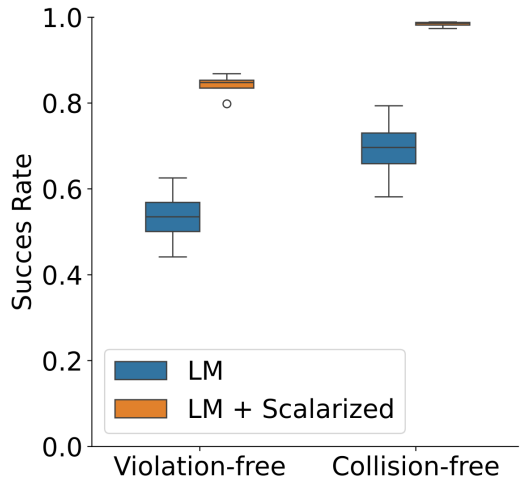

Figure 4. Success performance rate over violations and collisions.

The results show that the proposed approach yields better results without additional penalties on the reward function. This was achieved because the arm is better at avoiding the object to reach the goal state, yielding consistent steps. In addition, the trend in average rewards for the proposed approach slightly reduces before finding a constant average, indicating a more risk-averse stance as λ increases. However, the trend for the LM approach decreases with the number of steps and the inability to fully avoid the object when moving closer to the goal state.

5.4. Cost Limit Parameter Analysis

This subsection analyses the results derived from the cost limit variation and the performance of the proposed method with scalarized expectation and Lagrange multipliers (LM). The performance of the method over violations and collisions is shown in Figure 5. The range of the plots is represented by the minimum and maximum values across five seeds, with the solid line indicating the average.

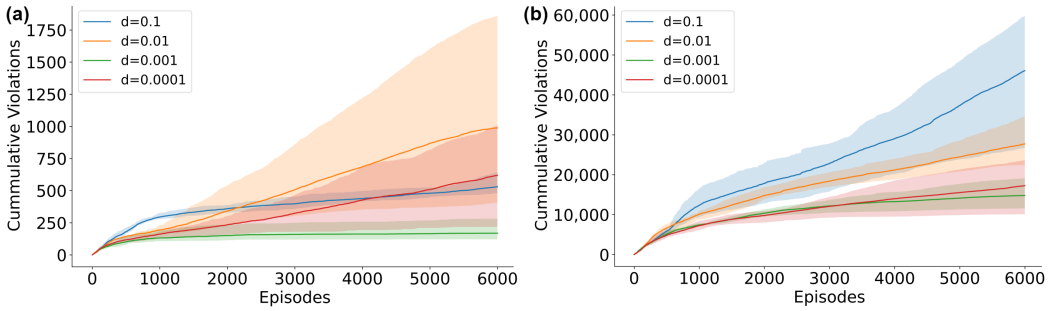

Figure 5. Effects of cost limit variation over (**a**) cumulative violations and (**b**) cumulative collisions. For visualization purposes, the number of episodes is reduced.

The cost limit, d, plays an important role in reducing violations and collisions when using the scalarized expected return with the LM approach. Figure 5 shows the effect, highlighting the relationship between d and the number of violations and collisions. A smaller value of d corresponds to fewer cumulative violations and collisions. This effect occurs because, as d decreases, the allowable cost decreases. Setting the cost limit $d = 0$ should theoretically reduce the violations to zero, at which point the reward function can be ignored.

5.5. Discussion

Combining scalarized expectation with Lagrange multipliers (LM) improves performance in avoiding obstacles in a dynamic environment, as seen in the UR5 setup. The method's effectiveness is evident through fewer violations and collisions, and a stable reward trend, compared with using LM alone. This suggests that the integrated approach better balances exploration and goal achievement, leading to more consistent and efficient task completion. The analysis also points out the role of cost limits in minimizing risks, indicating a direct correlation between lower cost limits and reduced incidence rates.

However, this approach has limitations, including its reliance on a deterministic indicator cost function and the challenges associated with min–max type problems in non-convex optimization. The first is the assumption that the indicator cost function is deterministic and relies on the labels of safe and unsafe regions being well defined. Stochasticity in the cost function can misguide the optimization objective towards optimizing in the wrong direction. In concrete terms, the cost function in areas labeled safe with some stochasticity would return $0 + \zeta$, where ζ can be inherent noise. This may switch the polarity of the return value labeled safe if the noise has a negative value, or could increase the return value, which directly affects Equation (10) because it relies on the immediate cost value. To alleviate this, a threshold could be established; however, the solution then becomes reliant on the threshold. The second caveat is the inherent challenges in min–max type problems in non-convex optimization [42]. A problem formulated in this setting can achieve only an approximation to the true solution because the max over policy π in Equation (4) is approximated with a neural network. This makes stability in training a problem that needs to be addressed. However, with direct scaling of the immediate cost function values, the challenges of min–max are reduced because the toggling effect treats it one objective at a time.

6. Conclusions and Future Work

This work highlights the challenges posed by robot–human coexistence and the importance of intuitive behavior from robots. A novel approach is presented by including a scalarized expected return in a CMDP problem and directly using the cost values sampled from the replay buffer. In addition to the trainable LM variable, which scales the expected cost, the cost samples used from the replay buffer scale the expected return and the expected cost as the SER scaling component. It is easier to interpret the workings of this approach on a single sample basis in a non-deep RL setting. The method optimizes expected returns in areas not designated with a cost label of 1, while actively avoiding expected costs in areas labeled with a cost of 1. This strategy is more straightforward than the iterative adjustment of λ to a point where the objective is balanced, specifically for avoiding areas with a cost label of 1. In addition, a notable limitation of using solely LM without the scalarized expectation is the need for a large number of samples to achieve the optimal balance. In contrast, the proposed approach achieves effective results. It advances the ability for robots to be present in shared spaces, paving the way for safer and more effective interactions between humans and machines in dynamic environments.

In future work, it would be valuable to deploy and test a model of the proposed approach on an actual UR5 robotic arm. Such a deployment would provide valuable insights into how the model is scalable and how effective the model is in real-world scenarios beyond simulated environments. In addition, exploring integration of the scalarization method with the coupled CMDP approach, which would be an extension of composite MDP as described by Singh et al. [43], could be a promising direction. In this expanded framework, each nonstationary link of the arm would be assigned its own constraint multiplier, leading to a multi-dimensional constraint problem represented by a vector of $\boldsymbol{\lambda} \in \mathbb{R}^{|L|}$, where $|L|$ signifies the number of links. This approach would effectively create a system of composite CMDPs, enabling more nuanced and link-specific optimization in the arm's movement and decision-making.

Author Contributions: Formal analysis, P.A.; funding acquisition, M.A.M.C.; investigation, P.A.; methodology, P.A.; resources, M.A.M.C.; software, P.A.; supervision, N.T., S.G.-R. and M.A.M.C.; validation, P.A., N.T., S.G.-R. and M.A.M.C.; visualization, P.A. and S.G.-R.; writing—original draft, P.A.; writing—review and editing, N.T., S.G.-R. and M.A.M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Sciences and Engineering Research Council: Alliance International grant ALLRP 576990—22.

Data Availability Statement: The data are contained within the article.

Acknowledgments: The authors would like to thank Luisa Liboni and INESC TEC laboratory in Porto for their contributions during the initial stages of this research project. P.A. thank the IBET Fellowship Program for the support received during his doctoral studies in Canada. S.G.-R. thanks the SENESCYT from Ecuador for the support received during his doctoral studies in Canada.

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

Abbreviations

The following abbreviations are used in this manuscript:

CMDP	Constrained Markov decision process
EE	End-effector
LM	Lagrange multiplier
MDP	Markov decision process
RL	Reinforcement learning
SAC	Soft actor–critic
SER	Scalarized expected return

References

1. Colgate, E.; Bicchi, A.; Peshkin, M.A.; Colgate, J.E. Safety for physical human-robot interaction. In *Springer Handbook of Robotics*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 1335–1348.
2. Beetz, M.; Chatila, R.; Hertzberg, J.; Pecora, F. *AI Reasoning Methods for Robotics*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 329–356.
3. Ingrand, F.; Ghallab, M. Deliberation for autonomous robots: A survey. *Artif. Intell.* **2017**, *247*, 10–44. Special Issue on AI and Robotics. [CrossRef]
4. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
5. Hasselt, H.V.; Guez, A.; Silver, D. Deep Reinforcement Learning with Double Q-Learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, Phoenix, AZ, USA, 12–17 February 2016; AAAI Press: Washington, DC, USA, 2016; pp. 2094–2100.
6. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
7. Gosavi, A. Reinforcement learning: A tutorial survey and recent advances. *INFORMS J. Comput.* **2009**, *21*, 178–192. [CrossRef]
8. Moldovan, T.M.; Abbeel, P. Safe Exploration in Markov Decision Processes. In Proceedings of the 29th International Coference on International Conference on Machine Learning, ICML'12, Madison, WI, USA, 26 June–1 July 2012; pp. 1451–1458.
9. Hans, A.; Schneegaß, D.; Schäfer, A.M.; Udluft, S. Safe exploration for reinforcement learning. In Proceedings of the ESANN, Bruges, Belgium, 23–25 April 2008; pp. 143–148.
10. Altman, E. *Constrained Markov Decision Processes*; CRC Press: Boca Raton, FL, USA, 1999; Volume 7.
11. Garcıa, J.; Fernández, F. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **2015**, *16*, 1437–1480.
12. Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; Mané, D. Concrete problems in AI safety. *arXiv* **2016**, arXiv:1606.06565.
13. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
14. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
15. Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI conference on artificial intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
16. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
17. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the 35th International Conference on Machine Learning (ICML-18), Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
18. Achiam, J.; Held, D.; Tamar, A.; Abbeel, P. Constrained policy optimization. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 22–31.
19. Bertsekas, D.P. *Constrained Optimization and Lagrange Multiplier Methods*; Academic Press: Cambridge, MA, USA, 2014.
20. Hayes, C.F.; Reymond, M.; Roijers, D.M.; Howley, E.; Mannion, P. Monte Carlo tree search algorithms for risk-aware and multi-objective reinforcement learning. *Auton. Agents -Multi-Agent Syst.* **2023**, *37*, 26. [CrossRef]
21. Jaimungal, S.; Pesenti, S.M.; Wang, Y.S.; Tatsat, H. Robust Risk-Aware Reinforcement Learning. *SIAM J. Financ. Math.* **2022**, *13*, 213–226. [CrossRef]
22. Geibel, P.; Wysotzki, F. Risk-Sensitive Reinforcement Learning Applied to Control under Constraints. *J. Artif. Int. Res.* **2005**, *24*, 81–108. [CrossRef]
23. Mihatsch, O.; Neuneier, R. Risk-sensitive reinforcement learning. *Mach. Learn.* **2002**, *49*, 267–290. [CrossRef]
24. Bossens, D.M.; Bishop, N. Explicit Explore, Exploit, or Escape (E 4): Near-optimal safety-constrained reinforcement learning in polynomial time. *Mach. Learn.* **2022**, *112*, 1–42. [CrossRef]
25. Wolff, E.M.; Topcu, U.; Murray, R.M. Robust control of uncertain Markov Decision Processes with temporal logic specifications. In Proceedings of the 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), Maui, HI, USA, 10–13 December 2012; pp. 3372–3379. [CrossRef]
26. Russel, R.H.; Benosman, M.; Van Baar, J. Robust constrained-MDPs: Soft-constrained robust policy optimization under model uncertainty. *arXiv* **2020**, arXiv:2010.04870.
27. Chen, B.; Liu, Z.; Zhu, J.; Xu, M.; Ding, W.; Li, L.; Zhao, D. Context-aware safe reinforcement learning for non-stationary environments. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xian, China, 30 May–5 June 2021; IEEE: Piscateville, NJ, USA, 2021; pp. 10689–10695.
28. Wachi, A.; Sui, Y. Safe reinforcement learning in constrained Markov decision processes. In Proceedings of the International Conference on Machine Learning. PMLR, Virtual, 13–18 July 2020; pp. 9797–9806.
29. Borkar, V.; Jain, R. Risk-constrained Markov decision processes. *IEEE Trans. Autom. Control.* **2014**, *59*, 2574–2579. [CrossRef]
30. Li, Y.; Hao, X.; She, Y.; Li, S.; Yu, M. Constrained motion planning of free-float dual-arm space manipulator via deep reinforcement learning. *Aerosp. Sci. Technol.* **2021**, *109*, 106446. [CrossRef]
31. Cao, Y.; Wang, S.; Zheng, X.; Ma, W.; Xie, X.; Liu, L. Reinforcement learning with prior policy guidance for motion planning of dual-arm free-floating space robot. *Aerosp. Sci. Technol.* **2023**, *136*, 108098. [CrossRef]

32. Li, Z.; Ma, H.; Ding, Y.; Wang, C.; Jin, Y. Motion Planning of Six-DOF Arm Robot Based on Improved DDPG Algorithm. In Proceedings of the 2020 39th Chinese Control Conference (CCC), Shenyang, China, 27–29 July 2020; pp. 3954–3959. [CrossRef]
33. Tang, W.; Cheng, C.; Ai, H.; Chen, L. Dual-Arm Robot Trajectory Planning Based on Deep Reinforcement Learning under Complex Environment. *Micromachines* **2022**, *13*, 564. [CrossRef] [PubMed]
34. Sangiovanni, B.; Rendiniello, A.; Incremona, G.P.; Ferrara, A.; Piastra, M. Deep reinforcement learning for collision avoidance of robotic manipulators. In Proceedings of the 2018 European Control Conference (ECC), Limassol, Cyprus, 12–15 June 2018; IEEE: Piscateville, NJ, USA, 2018; pp. 2063–2068.
35. Prianto, E.; Kim, M.; Park, J.H.; Bae, J.H.; Kim, J.S. Path Planning for Multi-Arm Manipulators Using Deep Reinforcement Learning: Soft Actor–Critic with Hindsight Experience Replay. *Sensors* **2020**, *20*, 5911. [CrossRef] [PubMed]
36. Zeng, R.; Liu, M.; Zhang, J.; Li, X.; Zhou, Q.; Jiang, Y. Manipulator Control Method Based on Deep Reinforcement Learning. In Proceedings of the 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 22–24 August 2020; IEEE: Piscateville, NJ, USA, 2020; pp. 415–420. [CrossRef]
37. Yang, S.; Wang, Q. Robotic Arm Motion Planning with Autonomous Obstacle Avoidance Based on Deep Reinforcement Learning. In Proceedings of the 2022 41st Chinese Control Conference (CCC), Heifei, China, 25–27 July 2022; pp. 3692–3697. [CrossRef]
38. Kamali, K.; Bonev, I.A.; Desrosiers, C. Real-time Motion Planning for Robotic Teleoperation Using Dynamic-goal Deep Reinforcement Learning. In Proceedings of the 2020 17th Conference on Computer and Robot Vision (CRV), Ottawa, ON, Canada, 13–15 May 2020; pp. 182–189. [CrossRef]
39. Avaei, A.; van der Spaa, L.; Peternel, L.; Kober, J. An Incremental Inverse Reinforcement Learning Approach for Motion Planning with Separated Path and Velocity Preferences. *Robotics* **2023**, *12*, 61. [CrossRef]
40. Todorov, E.; Erez, T.; Tassa, Y. MuJoCo: A physics engine for model-based control. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Algarve, 7–12 October 2012; pp. 5026–5033. [CrossRef]
41. Ray, A.; Achiam, J.; Amodei, D. Benchmarking safe exploration in deep reinforcement learning. *arXiv* **2019**, arXiv:1910.01708.
42. Razaviyayn, M.; Huang, T.; Lu, S.; Nouiehed, M.; Sanjabi, M.; Hong, M. Nonconvex min-max optimization: Applications, challenges, and recent theoretical advances. *IEEE Signal Process. Mag.* **2020**, *37*, 55–66. [CrossRef]
43. Singh, S.; Cohn, D. How to dynamically merge Markov decision processes. *Adv. Neural Inf. Process. Syst.* **1997**, *10*, 1057–1063.