Situational-Constrained Sequential Resources Allocation via Reinforcement Learning

Libo Zhang[1,2] , **Yang Chen**[3] , **Toru Takisaka**[1] , **Kaiqi Zhao**[2] , **Weidong Li**[2] and **Jiamou Liu**[2]

[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China
[2]The University of Auckland
[3]The University of New South Wales

{lzha797,wli916}@aucklanduni.ac.nz, takisaka@uestc.edu.cn, {kaiqi.zhao,jiamou.liu}@auckland.ac.nz, yang.chen.csphd@gmail.com

Abstract

Sequential Resource Allocation with situational constraints presents a significant challenge in real-world applications, where resource demands and priorities are context-dependent. This paper introduces a novel framework, SCRL, to address this problem. We formalize situational constraints as logic implications and develop a new algorithm that dynamically penalizes constraint violations. To handle situational constraints effectively, we propose a probabilistic selection mechanism to overcome limitations of traditional constraint reinforcement learning (CRL) approaches. We evaluate SCRL across two scenarios: medical resource allocation during a pandemic and pesticide distribution in agriculture. Experiments demonstrate that SCRL outperforms existing baselines in satisfying constraints while maintaining high resource efficiency, showcasing its potential for real-world, context-sensitive decision-making tasks.

1 Introduction

Sequential resource allocation (SRA) involves distributing limited resources across locations over time, where an agent allocates resources at a sequence of demand nodes while satisfying upper and lower bound constraints. The objective is to allocate resources efficiently while adhering to these constraints. SRA arises in critical domains such as healthcare, public safety, energy, and agriculture, where dynamic demands and societal priorities play a key role. For example, healthcare resource distribution during pandemics must balance immediate needs with future demand [Malenica *et al.*, 2024], while pesticide distribution must adapt to regional crop health and sustainability requirements [Qin *et al.*, 2021].

Beyond efficiency, resource allocation algorithms must consider societal constraints, such as equity [Pu, 2021], sustainability [Heffron and McCauley, 2014], and justice [Zhao *et al.*, 2020]. Moreover, these constraints often depend on context, such as prioritizing equity when regions' demands conflict. For example, during the COVID-19 pandemic, New Zealand proposed a Traffic Light system [Taylor *et al.*,

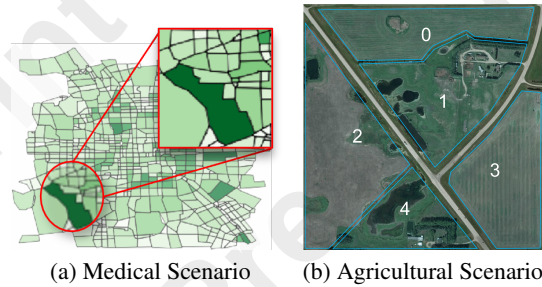(a) Medical Scenario   (b) Agricultural Scenario

Figure 1: **Left:** Simulated medical resource demand in Beijing, where darker colors represent higher demand levels. The experiment focuses on a district in the southwest. **Right:** Farmland in Saskatchewan, Canada, used for pesticide allocation. Numbers indicate regions with varying pesticide requirements.

2023] to adjust policies according to the level of emergency. E.g. one-meter distancing measures were only enforced when public medical resources faced high pressure. Similarly, the U.S. clean-energy supply-chain strategy [Igogo, 2022] highlighted adaptive systems to address bottlenecks during disruptions. They underscore the need for context-aware allocation strategies.

Traditional solutions to SRA, such as dynamic programming [Lien *et al.*, 2014] and multi-armed bandits [Kaufmann, 2018], are effective for small-scale problems with explicit models but struggle with scalability. Reinforcement learning (RL) offers a promising alternative by learning optimal policies through interaction with the environment, without requiring prior system knowledge [Bhatia *et al.*, 2019]. RL has been successfully applied to diverse SRA tasks, such as pesticide spraying [Qin *et al.*, 2021], healthcare resource allocation [Li *et al.*, 2023], and dynamic electricity distribution [Bahrami *et al.*, 2020]. Constrained reinforcement learning (CRL) extends standard RL by incorporating constraints into the learning objective, typically through Lagrangian methods or constrained policy updates. Recent advancements, such as density-constrained reinforcement learning (DCRL) [Qin *et al.*, 2021], have extended RL by incorporating constraints on state distributions. However, existing algorithms rely on static constraints, limiting their ability to

adapt to evolving demands and situational requirements. Addressing this gap calls for an advanced RL framework capable of incorporating conditional constraints to enable adaptive decision-making. This leads to the question: *How can we design a density-constrained RL framework that ensures situational fairness and adapts to dynamic, context-dependent constraints in resource allocation?* To address this question, one needs to (1) develop a formal framework for SRA under such constraints. (2) Propose a new density-constrained RL algorithm that handles "if-then" logic for such constraints.

In this paper, we initiate the study of situational constraints for SRA tasks. We formulate the problem as a conditional DCRL problem, where the constraints are implications. To address this problem, we propose a new algorithm, Situational-Constrained Reinforcement Learning (SCRL). The algorithm extends the conventional CRL framework by introducing a violation degree-based punitive term function that quantifies the extent of constraint violations and adjusts policy updates accordingly. Unlike previous approaches [Tessler *et al.*, 2018; Ray *et al.*, 2019; Qin *et al.*, 2021], SCRL incorporates an adaptive aggregation mechanism that handles disjunctive constraints by selectively prioritizing one constraint within the disjunction. This design allows SCRL to dynamically balance reward optimization with constraint satisfaction in context-sensitive environments. To the best of our knowledge, this is the first work to address situational, disjunctive constraints within the CRL paradigm.

We evaluate SCRL in two real-world-inspired scenarios: medical resource allocation during the COVID-19 pandemic in Beijing, China [Hao *et al.*, 2021] and agricultural resource distribution in Saskatchewan, Canada [Qin *et al.*, 2021]. In both cases, the constraints are designed to balance equity and adequacy, such as ensuring fairness when resources are insufficient and maintaining sufficient coverage when resources are ample. Experimental results demonstrate that SCRL significantly improves the satisfaction of situational constraints compared to baseline methods and effectively adapts resource distributions to meet context-specific requirements. Additionally, we present a case study to illustrate how SCRL adjusts resource allocation across regions in response to shifting situational demands, further highlighting the algorithm's ability to provide adaptive and equitable decision-making in complex, real-world environments.

The following is a summary of key contributions:

- Formulation of sequential resource allocation with situational, disjunctive constraints.

- Development of the SCRL algorithm with a violation degree-based punitive term for dynamic policy updates.

- Introduction of an aggregation mechanism to handle disjunctive constraints in context-sensitive environments.

2 Related Work

Sequential Resource Allocation (SRA). SRA focuses on distributing resources in systems where demands arrive sequentially, making it distinct from traditional resource allocation due to its dynamic nature and uncertainty. Its relevance spans socially impactful applications, such as allocating medical testing resources during pandemics [Malenica *et*

al., 2024] and optimizing industrial gas deliveries to minimize costs and prevent shortages [Berman and Larson, 2001]. Ethical considerations, such as equity in resource distribution, have also been explored in government and community planning [Johnson and Smilowitz, 2007].

Early approaches to SRA used dynamic programming to optimize costs under uncertainty, including supply chain management for sequential customers [Bassok and Ernst, 1995]. Bayesian methods were later introduced to handle stochastic dynamics, with Bayes-UCB demonstrating asymptotic optimality [Kaufmann, 2018]. To address fairness, heuristic algorithms were proposed for equitable and sustainable allocation [Lien *et al.*, 2014]. Reinforcement learning (RL) has recently become a predominant approach for SRA, offering scalable solutions for complex environments. Deep RL has been applied to supply chain management [Peng *et al.*, 2019] and network slicing [Liu *et al.*, 2021], enabling efficient resource allocation under constraints. Resource-constrained RL frameworks have further improved performance over conventional policies [Bhatia *et al.*, 2019].

Despite these advancements, existing methods often focus on fixed constraints or single-objective optimization, leaving a gap in addressing situational and context-sensitive constraints, which are critical for real-world applications.

Constrained reinforcement learning (CRL). CRL extends traditional RL by incorporating constraints to ensure policies satisfy predefined requirements while maximizing rewards [Gu *et al.*, 2022; García and Fernández, 2015]. Rooted in Constrained Markov Decision Processes (CMDPs)[Altman, 1993], methods like Reward Constrained Policy Optimization (RCPO)[Tessler *et al.*, 2018] and SAC-Lag [Ha *et al.*, 2020] use Lagrange multipliers to balance rewards and constraints. Constrained Policy Optimization (CPO)[Achiam *et al.*, 2017] introduced trust region methods for maintaining feasibility during updates, while Projection-based CPO (PCPO)[Yang *et al.*, 2020] extended it to further avoid infeasible policy during optimization.

Density-based CRL imposes constraints directly on state density functions, offering clear physical interpretations suitable for resource and safety-critical applications [Rantzer, 2001]. Qin et al.[Qin *et al.*, 2021] applied this approach to a pesticide spraying scenario by constraining pesticide density, and Zhang et al.[Zhang *et al.*, 2023] extended it to multi-agent settings with ethical constraints. These methods demonstrated the efficacy of density constraints for resource allocation but do not address situational constraints, which require dynamic adaptation across scenarios.

Other RL paradigms are less applicable to SRA. Logic-based RL [Hasanbeig *et al.*, 2018; Hasanbeig *et al.*, 2020] relies on qualitative specifications, which are unsuitable for quantitative resource allocation and introduce significant computational complexity. Fuzzy-logic-based RL, e.g., FQL [Glorennec and Jouffe, 1997], applies fuzzy rules to represent value functions and actions, and has been used in tasks like robot navigation [Fathinezhad *et al.*, 2016] and resource management [Prasath *et al.*, 2024]. However, it typically yields soft rule satisfaction, which is unsuitable for strict constraints like fairness and safety. Shielding [Waga *et al.*, 2022;

Alshiekh *et al.*, 2017] focuses on safe exploration, which is unnecessary for SRA with reliable simulations.

Our approach builds on density-based CRL, extending it to handle situational constraints that cannot be addressed by traditional Lagrangian methods. We propose a novel algorithm tailored to this problem, ensuring dynamic and context-sensitive resource allocation.

3 Problem Formulation

3.1 Situational Constraints

We study resource allocation to demand nodes represented as finite regions within a 2D spatial domain, reflecting applications where resources are distributed geographically. Formally, a supplier agent distributes resources over a bounded region $D \subseteq \mathbb{R}^2$ with $m \in \mathbb{N}$ demand nodes, indexed as $M = \{1, \ldots, m\}$. Each demand node $i \in M$ covers a sub-region $S_i \subseteq D$, and an allocation function $f \colon M \to \mathbb{R}$ maps each node i to a non-negative amount of resources $f(i) \geq 0$. These sub-regions may overlap. Applications include drones spraying pesticides [Qin *et al.*, 2021], mobile immunization vehicles distributing vaccines, and policing resource allocation [Maslen and Paine, 2024], as illustrated in Figure 1.

In many situations, resource allocation must satisfy interval and equity constraints. Interval constraints ensure that each demand node i receives resources within a specified range $f(i) \in [a_i, b_i]$, where $0 \leq a_i < b_i \leq \infty$ [Qin *et al.*, 2021]. Equity constraints ensure fairness, expressed as $|f(i) - f(j)| \leq b$, where $b \geq 0$ [Lien *et al.*, 2014; Zhang *et al.*, 2023]. These constraints address sufficiency and fairness but lack flexibility for context-aware allocation. We therefore introduce situational constraints, which enable conditional relationships between constraints. For example, a situational constraint may state: "If the resources allocated to region A exceed a certain threshold, then region B must receive a minimum amount." Let $\vec{f} = [f(1), \ldots, f(m)]$ represent the allocation vector.

Definition 1. *An* atomic constraint *is of the form* $\vec{a} \cdot \vec{f} \leq b$ *where vector* $\vec{a} \in \mathbb{R}^m$ *and* $b \in \mathbb{R}$ *are parameters. A* situational constraint *is of the form* $\varphi_1(\vec{f}) \to \varphi_2(\vec{f})$, *where* $\varphi_1(\vec{f})$ *and* $\varphi_2(\vec{f})$ *are atomic.*

Situational constraints generalize existing formulations. For example, an interval constraint $f(i) \in [a_i, b_i]$ can be expressed as $\top \to [-f(i) \leq -a_i]$ and $\top \to [f(i) \leq b_i]$, where $\top$ denotes an always-true atomic constraint. Similarly, an equity constraint can be rewritten as a conjunction of two interval constraints. While interval and equity constraints have been studied [Qin *et al.*, 2021; Lien *et al.*, 2014; Zhang *et al.*, 2023], they fail to capture context-sensitive requirements. Situational constraints address this gap by capturing conditional demands.

3.2 SRA with Situational Constraints

Formally, the SRA problem is represented by the MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, r, P, \gamma, \eta, L \rangle$, where:

- $\mathcal{S}$: The state space, which captures the agent's position in the 2D region and its movement dynamics, such as velocity.

- $\mathcal{A}$: The action space, which represents the set of actions available to the agent.

- $P \colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$: The transition function, which defines the probability distribution over next states s' given the current state s and action a.

- $r \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: The reward function, which quantifies the efficiency of resource allocation.

- $\gamma \in (0, 1)$: The discount factor.

- $\eta \in \Delta(\mathcal{S})$: The initial state distribution, which specifies the probability distribution over initial states $s_0 \in \mathcal{S}$.

- $L \colon \mathcal{S} \to 2^M$: The labeling function maps $s \in \mathcal{S}$ to the set of demand nodes $L(s)$ receiving resources at that state.

To formalize the reward function r, we follow [Qin *et al.*, 2021] and assume that the agent distributes resources at a constant rate as it moves through the region. This means that the amount of resources $f(i)$ allocated to a demand node $i \in M$ is captured by the time the agent spends within the corresponding sub-region S_i. This simplification links resource allocation directly to the agent's trajectory, allowing the agent to control resource distribution through its movement across the space. Formally, a *trajectory* is a potentially infinite sequence $\tau = (s_0, s_1, s_2, \ldots)$ where each $s_t \in \mathcal{S}$ represents the agent's state at time t. The following definition follows the definition in [Qin *et al.*, 2021; Rantzer, 2001; Syed *et al.*, 2008; Chen and Ames, 2019].

Definition 2. *Given a trajectory* τ, *the* density *of resources allocated to demand node* $i \in M$ *is defined as:* $\rho^\tau(i) := \sum_{t=0}^{\infty} \gamma^t \cdot \mathbb{1}_{L(s_t)}(i)$, *where* $\mathbb{1}$ *is the indicator function.*

The density $\rho^\tau(i)$ quantifies the cumulative, discounted amount of resources allocated to demand node i by the agent while following the trajectory τ. The discount factor ensures the density function does not diverge under infinite-horizon settings. A *policy* $\pi \colon \mathcal{S} \to \Delta(\mathcal{A})$ defines a probability distribution over the agent's actions at each state. The sequence of states $\tau = (s_0, s_1, \ldots)$ *conforms to* a policy π, written $\tau \sim \pi$, if: $s_{t+1} \sim P(s_t, a_t)$, $a_t \sim \pi(s_t)$ for all $t \geq 0$, where $P(s_t, a_t)$ is the transition function.

Definition 3. *Given a policy* $\pi \colon \mathcal{S} \to \Delta(\mathcal{A})$, *the* expected density *of resources allocated to demand node* $i \in M$ *is:*

$$\rho^\pi(i) := \mathbb{E}_{\tau \sim \pi}[\rho^\tau(i)] = \sum_{t=0}^{\infty} \gamma^t \Pr(i \in L(s_t) \mid \pi, \eta),$$

where $\Pr(i \in L(s_t) \mid \pi, \eta)$ *denotes the probability that demand node* i *is receiving resources at time* t, *given the policy* π *and initial state distribution* η.

The expected density $\rho^\pi(i)$ captures the average amount of resources allocated to demand node i when the agent follows policy π. Thus any atomic constraint $\varphi(\vec{f})$ of the form $\vec{a} \cdot \vec{f} \leq b$ can be rephrased as the following constraint over the policy π of the agent: $a_1 \rho^\pi(1) + \cdots + a_m \rho^\pi(m) \leq b$. We now formalize our main problem:

Problem 1 (SRA with situational constraints).

$$\arg\max_\pi \sum_{s \in \mathcal{S}} \eta(s) V_\pi(s) \qquad s.t. \qquad \Psi(\pi)$$

where $V_\pi(s)$ is the expected cumulative reward under policy π, defined as: $V_\pi(s) = \mathbb{E}_{\tau \sim \pi, s_0 = s}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, and $\Psi(\pi)$ is a conjunction of situational constraints over π.

4 Method

4.1 Algorithm Overview

To solve Problem 1, we propose a general algorithm framework (Alg. 1) designed for constrained reinforcement learning (CRL) tasks. The framework dynamically addresses the dual objectives of maximizing cumulative rewards and satisfying constraints by incorporating a *punitive mechanism* through a *punitive term function*, denoted as $\sigma(\Psi, s)$. This term penalizes constraint violations directly within the reward structure, effectively transforming the constrained RL problem into an unconstrained optimization task. The objective then becomes maximizing the cumulative punished reward, allowing the agent to learn constraint satisfaction implicitly while pursuing reward optimization.

The framework operates iteratively through three core steps: (1) **Trajectory generation:** The current policy π is used to generate a collection of trajectories D_π. (2) **Constraint evaluation and punitive term update:** Using D_π, the violation degree $Vio^\pi(\Psi)$ is computed to evaluate how well the policy satisfies the constraints, and $\sigma(\Psi, s)$ is updated accordingly. (3) **Policy optimization:** The policy π is updated by maximizing the cumulative punished reward. This iterative process continues until convergence, ensuring that both reward maximization and constraint satisfaction objectives are met. This framework generalizes existing CRL methods, including RCPO [Tessler *et al.*, 2018], PPO-Lag [Ray *et al.*, 2019], and DCRL [Qin *et al.*, 2021], which utilize scalar Lagrangian multipliers as punitive terms.

We extend the punitive mechanism to handle situational constraints. This involves: 1. Defining the punitive term $\sigma(\varphi, s)$ for atomic constraints, ensuring it accurately reflects the violation degree. 2. Extending σ to situational constraints, such as $\sigma(\varphi_1 \rightarrow \varphi_2, s)$.

4.2 Punitive Mechanism: Atomic Constraints

For an atomic constraint φ of the form $\vec{a} \cdot \vec{\rho^\pi} \leq b$, the violation degree is defined as: $Vio^\pi(\varphi) := \vec{a} \cdot \vec{\rho^\pi} - b$. This quantifies the extent to which the constraint φ is violated under the current

---

Algorithm 1 Algorithm Scheme with Punitive Term

1: **Input:** An MDP $\mathcal{M}$ with a constraint Ψ
2: **Initialize:** An initial policy π; A punitive term σ
3: **while** Not converged **do**
4:     **Generate trajectories** $D_\pi \leftarrow \{\tau_1, \tau_2, \cdots \mid \eta, \pi, P\}$
5:     **Evaluate violation degree** $Vio^\pi(\Psi)$ using D_π
6:     **Update punitive term function** $\sigma(\Psi, s)$
7:     **for** each transition $(s, a, r, s') \in \tau_i$ where $\tau_i \in D_\pi$ **do**
8:         Apply punitive term on reward $r' \leftarrow r - \sigma(\Psi, s)$
9:     **end for**
10:    **Update policy** $\pi \leftarrow \arg\max_\pi \mathbb{E}_{D_\pi}\left[\sum_t \gamma^t r'(s_t, a_t)\right]$
11: **end while**
12: Return π

---

policy π. A positive $Vio^\pi(\varphi)$ indicates a violation, while a value of zero or less signifies satisfaction of the constraint.

To design a punitive mechanism for atomic constraints, we decompose it into two complementary components: the *penalty factor*, which measures the overall severity of a constraint violation, and the *weighting factor*, which determines the state-level impact of the violation. These components collectively define the punitive term $\sigma(\varphi, s)$.

1. Penalty factor: The penalty factor $\kappa(\varphi)$ is updated iteratively to reflect the accumulated violation of φ over time:

$$\kappa'(\varphi) := \max(0, \kappa(\varphi) + \beta \cdot Vio^\pi(\varphi)),$$

where β is the learning rate. This approach aligns with existing CRL methods, such as DCRL [Qin *et al.*, 2021], where the penalty factor is dynamically adjusted to enforce density constraints. For instance, DCRL enforces an upper bound ρ_{max} on state density $\rho^\pi(s')$ by updating κ as $\kappa' = \max(0, \kappa + \beta \cdot (\rho^\pi(s') - \rho_{max}))$. Our mechanism generalizes this idea to arbitrary atomic constraints.

2. Weighting Factor: The weighting factor $w(\varphi, s)$ accounts for the fact that visiting different states may contribute unequally to the violation of φ by setting

$$w(\varphi, s) := \frac{\sum_{i=1}^m a_i \cdot \mathbb{1}_{L(s)}(i)}{\sum_{i=1}^m |a_i|},$$

where $\mathbb{1}$ is the indicator function, and $L(s)$ identifies the demand nodes affected by state s. Thus the weighting factor $w(\varphi, s)$ reflects how allocating resources to state s impacts the violation degree $Vio^\pi(\varphi)$.

3. Punitive term function: The punitive term for an atomic constraint φ is defined as: $\sigma(\varphi, s) = w(\varphi, s) \cdot \kappa(\varphi)$. The penalized reward is then computed as:

$$r'(s, a) = r(s, a) - \sigma(\varphi, s).$$

This formulation ensures that the agent is guided toward satisfying the constraint by dynamically adjusting the reward based on the impact of each state on the violation degree.

Example 1. *Consider atomic constraint φ that specifies $\rho^\pi(i) - \rho^\pi(j) \leq 0$, which enforces that the resources allocated to demand node i should not exceed those allocated to demand node j. Suppose that the penalty factor $\kappa(\varphi) > 0$, i.e., the constraint is violated under the current π. For $s \in \mathcal{S}$, when $i \in L(s)$ and $j \notin L(s)$, by definition, we have $w(\varphi, s) = 1/2$. In this case, $\sigma(\varphi, s) = 1/2\kappa(\varphi) > 0$. This leads to a decrease on the reward, which discourages the agent to allocate resource at s. On the other hand, when $i \notin L(s)$ and $j \in L(s)$, $w(\varphi, s) = -1/2$, which means $\sigma(\varphi, s) < 0$, leading to an increase on the reward, which encourages the agent to allocate resources at s. In any other case, $w(\varphi, s) = 0$, which means that any penalty at state s will not affect the satisfaction of φ.*

With the definition of the punitive term $\sigma(\varphi, s)$ provided above, we can directly instantiate Algorithm 1 for addressing an SRA problem with an atomic constraint φ.

Proposition 1. *Consider an SRA problem with an atomic constraint φ of the form $\vec{a} \cdot \vec{\rho}^{\pi} \leq b$. With a sufficiently small learning rate β, the algorithm framework incorporating the punitive term $\sigma(\varphi, s)$ converges to a feasible solution.*

Proof. The proof builds on principles from canonical Lagrangian-based CRL frameworks, such as [Tessler *et al.*, 2018]. □

4.3 Punitive Mechanism: Situational Constraints

Situational constraints $\psi := \varphi_1 \rightarrow \varphi_2$ can be reformulated as disjunctive constraints $\neg\varphi_1 \vee \varphi_2$. This requires handling disjunctions of atomic constraints $\varphi_1 \vee \varphi_2$, which pose unique challenges. Traditional CRL methods work well for conjunctive constraints as they enable gradient-based optimization within connected feasible regions. However, disjunctive constraints create disconnected feasible regions that make gradient-based methods ineffective.

Conventional approaches usually reformulate disjunctive constraints into mixed-integer linear programming (MILP) [Trespalacios and Grossmann, 2015; Kronqvist *et al.*, 2021] and solve them using techniques such as branch-and-bound [Türkay and Grossmann, 1996]. While effective for problems with explicit system models (e.g., linear programming), these methods struggle with the complexity and dynamic nature of realistic tasks like the SRA problem, where constraints are context-dependent and the environment evolves sequentially. Moreover, branch-and-bound approaches scale poorly in large or high-dimensional problems due to their exhaustive exploration of disjuncts. Importantly, recent machine learning methods have tackled disjunctive constraints by interpreting them as a min-operator over loss functions or as unions of feasible sets [Ren *et al.*, 2020; Huang *et al.*, 2022; Li and Srikumar, 2019; Nandwani *et al.*, 2019]. Building on these ideas, our approach extends the use of min-operators to RL by designing a punitive mechanism for disjunctive constraints.

To define the punitive term $\sigma(\psi, s)$ on a state s, we adopt the principle that prioritizes the "least-violated" disjunct in the constraint, i.e., φ_j where $j = \arg\min_j\{\sigma(\varphi_1, s), \sigma(\varphi_2, s), \ldots, \sigma(\varphi_j, s)\}$. The min operator aligns with the logical semantics of disjunctions and has been applied in prior works [Ren *et al.*, 2020; Huang *et al.*, 2022]. This design encourages the policy to satisfy the most attainable disjunct in a disjunction.

While it is intuitive to select the least-violated atomic constraint during the optimization, greedily applying the min-operator may suffer from sub-optimality in cases where it focuses on an infeasible disjunct that appears easier to satisfy. Specifically, consider a disjunction $\psi := \varphi_1 \vee \varphi_2$, where φ_1 is infeasible for the policy set Π, meaning $\forall \pi \in \Pi$, $Vio^{\pi}(\varphi_1) > 0$, while φ_2 is feasible. In this scenario, for a given policy $\pi' \in \Pi$ at state s, if the punitive term for the infeasible disjunct φ_1, $\sigma(\varphi_1, s)$, is smaller than that for the feasible disjunct φ_2, $\sigma(\varphi_2, s)$, the algorithm would encourage the policy to prioritize φ_1. Consequently, the policy is misguided to focus on an unattainable constraint.

To address this issue, we propose a probabilistic mechanism for selecting disjuncts within a disjunctive constraint.

Algorithm 2 Situational-Constrained RL

1: **Input:** An MDP $(\mathcal{S}, \mathcal{A}, P, r, \eta, \gamma, L)$, situational constraints $\Psi := \bigwedge_{i \in I} \bigvee_{j \in \{1,2\}} \varphi_j$
2: **Initialisation:** Let π be a random policy, $\kappa(\varphi) = 0$ be penalty factor for each φ, β be learning rate for κ
3: **repeat**
4:     Generate trajectories $D_{\pi} = \{\tau_1, \tau_2, \cdots \mid \eta, \pi, P\}$
5:     Empirically compute density ρ^{π} according to D_{π}
6:     **for all** atomic constraint φ **do**
7:         Compute the violation degree $Vio^{\pi}(\varphi)$ for φ
8:         Update penalty factor as:
        $\kappa(\varphi) \leftarrow \max(0, \kappa(\varphi) + \beta Vio^{\pi}(\varphi))$
9:     **end for**
10:     **for** each $\tau_i \in D_{\pi}$, each transition $(s, a, r, s') \in \tau_i$ **do**
11:         Calculate $\sigma(\Psi, s)$.
12:         Apply punitive term on reward $r' \leftarrow r - \sigma(\Psi, s)$
13:     **end for**
14:     Solve π that maximizes the expected punished return based on D_{π}
15: **until** Convergence
16: **Output:** A policy π, with density values ρ^{π}.

For a disjunction $\psi = \bigvee_{j \in [J]} \varphi_j$, where φ_j are atomic constraints, probabilistic mechanism defines a random variable Φ over the set of atomic constraints. The punitive term $\sigma(\psi, s)$ is then defined as $\sigma(\Phi, s)$, where Φ follows a probability distribution that assigns a probability p_j to each φ_j. These probabilities are defined as $p_j := \frac{\kappa(\varphi_j)^{-1}}{\sum_{j \in J} \kappa(\varphi_j)^{-1}}$, where $\kappa(\varphi_j)$ is the penalty factor associated with φ_j. This formulation ensures that constraints closer to satisfaction are prioritized, while constraints with larger penalty factors are still occasionally explored due to their nonzero probabilities.

Finally, for a conjunction of I situational constraints, $\Psi := \wedge_{i \in I} \varphi_i$, we define $\sigma(\Psi, s) = \sum_{i \in I} \sigma(\psi_i, s)$.

4.4 SCRL Algorithm

Alg.2 presents the Situational-Constrained Reinforcement Learning (SCRL) algorithm, an instance of the general framework (Alg.1) tailored for Problem 1. SCRL incorporates the punitive mechanism defined above to handle situational constraints. The algorithm initializes the penalty factor $\kappa(\varphi) = 0$ for each atomic constraint φ, with a learning rate β to encourage exploration during the early stages of training. Trajectory data is used to empirically estimate the state density $\rho^{\pi}(s)$, leveraging either discrete state counts or kernel-based methods for continuous spaces [Qin *et al.*, 2021; Chen, 2017], ensuring computational efficiency for large-scale problems. The algorithm iteratively alternates between generating trajectories, updating penalty factors based on constraint violations, applying punitive terms to the rewards, and optimizing the policy. This iterative process ensures both constraint satisfaction and reward maximization.

5 Experiment

We aim to validate SCRL algorithm's performance through empirical evaluations over two real-world scenarios.
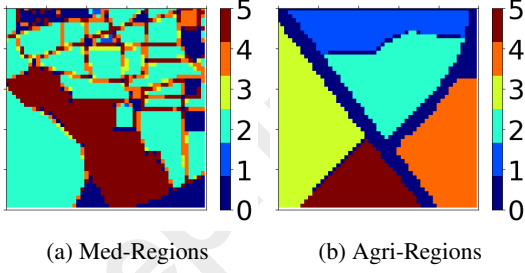
(a) Med-Regions   (b) Agri-Regions

Figure 2: The sub-regions in two scenarios. For both scenarios, the map is divided into 50×50 grids, and 5 regions (regions with label 0 are ignored in our settings).

5.1 Experiment Scenarios and Tasks

Medical Resource Allocation. This scenario models the allocation of medical resources in Beijing during the COVID-19 pandemic using a simulation from [Hao *et al.*, 2021]. The city is divided into modules, grouped into five sub-regions based on demand levels (Figure 2a). The challenge is to prioritize high-demand regions during resource shortages while maintaining fairness, reflecting real-world public health requirements for dynamic, context-sensitive allocation policies.

Agricultural Spraying Drone. [Qin *et al.*, 2021] This scenario involves pesticide allocation in farmland in Saskatchewan, Canada, divided into five sub-regions based on crop types (Figure 2b). The agent must optimize pesticide usage by responding to pest outbreaks while avoiding overuse, balancing sufficiency and fairness across regions. This mirrors real-world agricultural challenges with economic and environmental implications.

For both scenarios, the agent's goal is to maximize resource allocation efficiency while satisfying constraints. Three tasks of increasing complexity evaluate the agent's performance: 1. **Situational Task:** A single situational constraint requires, e.g., "If resources allocated to certain regions exceed a threshold, others must receive a minimum allocation." This tests the agent's ability to adapt dynamically to conditional requirements. 2. **Priority Task:** Involves equity constraints (equal resource allocation across specific regions) and adequacy constraints (minimum resources for specific regions). The situational requirement states, "If adequacy cannot be met, ensure equity." This evaluates the agent's ability to prioritize fairness under resource limitations. 3. **Joint Task:** Combines adequacy and equity constraints simultaneously without prioritization, requiring the agent to balance potentially conflicting requirements. In some cases, satisfying both constraints may be infeasible.

5.2 Baselines

We compare our approach against four baseline methods:

1. Deep Deterministic Policy Gradient (**DDPG**) [Gu *et al.*, 2017] serves as an unconstrained RL baseline, optimizing solely for reward without considering constraints.

2. Reward Constrained Policy Optimization (**RCPO**) [Tessler *et al.*, 2018] is adapted to our setting by defining cost functions over state-action pairs

Scenario	Task	Alg.	Cons.Vio	Reward
Med.	**Situational**	DDPG	4.84±0.28	**-2.09±0.01**
		RCPO	5.07±0.33	**-2.09±0.01**
		CAL	16.84±13.13	-4.05±2.16
		DCRL1	**0.0±0.0**	-14.65±1.02
		DCRL2	10.15±9.17	-11.65±13.94
		SCRL	**0.0±0.0**	-13.12±3.86
	Priority	DDPG	30.16±10.61	-2.74±2.03
		RCPO	33.36±0.75	**-2.1±0.01**
		CAL	13.31±7.37	-4.67±1.90
		DCRL1	32.15±4.44	-3.38±4.06
		DCRL2	7.95±14.14	-29.77±19.75
		SCRL	**0.0±0.0**	-18.52±13.17
	Joint	DDPG	54.04±0.23	**-2.09±0.01**
		RCPO	54.04±0.22	-2.1±0.01
		CAL	**36.18±0.4**	-5.02±3.02
		DCRL	41.03±7.9	-29.97±18.16
		SCRL	37.17±8.85	-9.62±4.17
Agri.	**Situational**	DDPG	4.85±0.17	**-2.1±0.0**
		RCPO	11.49±1.55	-3.2±0.3
		CAL	6.71±3.95	-3.79±3.34
		DCRL1	3.52±4.76	-13.69±3.15
		DCRL2	4.69±4.93	-19.15±14.76
		SCRL	**0.01±0.03**	-45.37±9.76
	Priority	DDPG	10.37±0.3	**-2.1±0.02**
		RCPO	1.26±0.67	-3.22±0.18
		CAL	7.64±2.35	-3.74±3.36
		DCRL1	**0.0±0.0**	-31.29±16.37
		DCRL2	3.88±4.9	-15.22±19.15
		SCRL	**0.0±0.0**	-9.46±1.20
	Joint	DDPG	19.54±0.15	**-2.1±0.01**
		RCPO	13.22±0.92	-3.04±0.21
		CAL	18.61±1.23	-2.39±0.23
		DCRL	20.3±10.66	-24±11.82
		SCRL	**4.31±2.38**	-24.53±5.07

Table 1: Result on two scenarios, each with three tasks. Mean and Std are collected from 10 independent runs.

instead of using density-based constraints. RCPO cannot natively support density-based situational constraints, requiring approximations for implementation.

3. Conservative Augmented Lagrangian (**CAL**) [Wu *et al.*, 2024] is a recent primal-dual CRL method. We adapt CAL to our setting in the same manner as RCPO.

4. Density Constrained Reinforcement Learning (**DCRL**) [Qin *et al.*, 2021] is included as a baseline but lacks native support for situational constraints. To adapt, we decompose each situational constraint $\varphi_1 \to \varphi_2$ into: **(DCRL1):** The premise $\neg\varphi_1$, treated as an interval constraint $\rho^{\pi}(s) \in [a, b]$. **(DCRL2):** The conclusion φ_2, addressed independently as an interval constraint.

5.3 Performance Metrics

The primary evaluation metric, **constraint violation (Cons.Vio.)**, measures violation of constraints. A lower Cons.Vio. indicates better compliance with these critical constraints and is preferred, as we prioritize the safety and fairness in real-world applications. The density function in experiments takes an undiscounted sum due to the finite-horizon setting. The secondary metric, **reward**, assesses resource efficiency, with higher rewards indicating less resource allocation amount. A negative reward is assigned when one unit of resources is allocated.

Scenario	Task	Alg.	Equity	Adequacy
Med	Situational	DDPG	**7.0**	125.1
		SCRL	24.8	**0.0**
	Joint	DDPG	**7.0**	50.0
		SCRL	9.6	**23.1**
Agri	Situational	DDPG	**6.9**	22.9
		SCRL	15.7	**0.0**
	Joint	DDPG	6.9	22.9
		SCRL	**3.8**	**0.0**

Table 2: Violation of different constraints. Violation of two constraints (equity and adequacy) in different tasks are shown. In Med. scenario, the adequacy requirement for two tasks are different, thus DDPG has different performances.
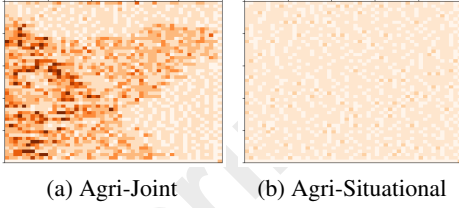

(a) Agri-Joint (b) Agri-Situational

Figure 3: SCRL's resource allocation as heatmap. Higher temprerature indicates more resources allocated. In joint task, agent tries to satisfy the equity constraint so allocating resources to different regions in different amounts; In situational task, agent prioritizes adequacy constraint so allocating each region with sufficient resources.

5.4 Experiment Result

The result in Table 1 shows that DDPG attains high rewards at the cost of severe constraint violations. Both RCPO and CAL rely on a surrogate cost function, so they suffer high violations (with the lone exception of CAL on the Med. joint task). Few DCRL instances satisfy constraints due to the separate implementation. However, DCRL lacks scalability against situational constraints. SCRL consistently offers near-zero cost on priority and situational tasks. For joint task, SCRL also offers low-level costs, demonstrating its advantage in satisfying context-sensitive requirements. Besides, even when a few DCRL instances satisfy constraints, they perform worse on rewards (e.g., -31 vs. -9). In contrast, SCRL successfully guides the agent to higher-reward feasible solutions.

5.5 Case Study: Priority and Joint Tasks

We investigate SCRL agent's behavior by comparing it with DDPG agent's behavior, as an unconstrained baseline.

Constraint Violation. As shown in Table 2, SCRL effectively prioritizes adequacy constraints in priority tasks, satisfying them at the expense of higher equity violations. In joint tasks, SCRL balances both adequacy and equity constraints, minimizing total violation degrees. This demonstrates SCRL's adaptability to diverse constraint structures.

Resources Allocation. Figure 3 visualizes SCRL's allocation strategies on Agri. scenario, with warmer colors indicating higher allocations. In joint tasks, SCRL balances equity constraints, reflecting region size variations. In priority tasks, it emphasizes adequacy, ensuring critical regions meet mini-

Scenario	Algorithm	Cons.Vio	Reward
Multi Disjunct	DDPG	8.6	**-2.1**
	RCPO	10.8	-4.2
	CAL	10	-2.3
	DCRL1	9.3	**-2.1**
	DCRL2	9.4	-14.2
	DCRL3	0	-16.7
	DCRL4	0	-48.3
	SCRL	0	-11.8

Table 3: Case study result on multi-disjunction task, Agri. scenario. The constraint involves four disjunctions.

Scenario	Alg.	Cons.Vio	Reward
Med.	SCRL-min	0.7 ± 2.3	**-16.0 ± 12.0**
	SCRL	**0.0 ± 0.0**	-18.5 ± 13.2
Agri.	SCRL-min	0.3 ± 0.8	-13.6 ± 8.2
	SCRL	**0.0 ± 0.0**	**-9.5 ± 1.2**

Table 4: Ablation study on probabilistic factor. We show the results on situational tasks in two scenarios. **SCRL-min** replaces probabilistic factor with min-operator, as defined in Sec. 4.3.

mum demands. These results demonstrate SCRL's adaptability to dynamically meet task-specific constraints.

Resource Efficiency. In the Agri. scenario, DDPG minimizes resource usage, completing its trajectory in 800 time steps, achieving higher rewards at the cost of constraint violations. In contrast, SCRL meets adequacy constraints, requiring at least 900 time steps to satisfy minimum demands across regions, leading to lower rewards. A similar trade-off is observed in the Med. scenario. This underscores the inherent trade-off between reward maximization and constraint satisfaction, as CRL algorithms prioritize constraint compliance over unconstrained efficiency.

5.6 Case Study: More disjunctions

Case study in Table. 3 shows algorithms performance under multiple disjunctions. The result is consistent with earlier analysis: RCPO and CAL fail due to the surrogate constraint; DCRL occasionally satisfies the constraint with a loss of reward (DCRL 3 and 4); SCRL still shows its advantage in capturing situational constraints.

5.7 Ablation Study: Probabilistic Mechanism

We compare the proposed probabilistic mechanism (SCRL) with a variant using the min operator (SCRL-min). As shown in Table 4, SCRL-min performs similarly in reward but incurs slightly higher constraint violations. This aligns with our earlier claim that the min operator may mislead the agent to satisfy an infeasible constraint with a lower violation.

6 Conclusion and Limitations

This paper tackles sequential resource allocation (SRA) under situational constraints, formulating the problem as an MDP with density functions to quantify regional resource allocation. We propose the SCRL algorithm, which trains agents under context-sensitive constraints.

Limitations. This study focuses on a single resource allocator and single-resource type. Extending the framework to multi-agent systems and multi-resource scenarios offers promising directions for future work.

References

[Achiam *et al.*, 2017] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

[Alshiekh *et al.*, 2017] Mohammed Alshiekh, Roderick Bloem, Ruediger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding, 2017.

[Altman, 1993] Eitan Altman. Asymptotic properties of constrained markov decision processes. *Zeitschrift für Operations Research*, 37(2):151–170, 1993.

[Bahrami *et al.*, 2020] Shahab Bahrami, Yu Christine Chen, and Vincent WS Wong. Deep reinforcement learning for demand response in distribution networks. *IEEE Transactions on Smart Grid*, 12(2):1496–1506, 2020.

[Bassok and Ernst, 1995] Yehuda Bassok and Ricardo Ernst. Dynamic allocations for multi-product distribution. *Transportation Science*, 29(3):256–266, 1995.

[Berman and Larson, 2001] Oded Berman and Richard C Larson. Deliveries in an inventory/routing problem using stochastic dynamic programming. *Transportation Science*, 35(2):192–213, 2001.

[Bhatia *et al.*, 2019] Abhinav Bhatia, Pradeep Varakantham, and Akshat Kumar. Resource constrained deep reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 610–620, 2019.

[Chen and Ames, 2019] Yuxiao Chen and Aaron D Ames. Duality between density function and value function with applications in constrained optimal control and markov decision process. *arXiv preprint arXiv:1902.09583*, 2019.

[Chen, 2017] Yen-Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1(1):161–187, 2017.

[Fathinezhad *et al.*, 2016] Fatemeh Fathinezhad, Vali Derhami, and Mehdi Rezaeian. Supervised fuzzy reinforcement learning for robot navigation. *Applied Soft Computing*, 40:33–41, 2016.

[Garcıa and Fernández, 2015] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[Glorennec and Jouffe, 1997] Pierre Yves Glorennec and Lionel Jouffe. Fuzzy q-learning. In *Proceedings of 6th international fuzzy systems conference*, volume 2, pages 659–662. IEEE, 1997.

[Gu *et al.*, 2017] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[Gu *et al.*, 2022] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.

[Ha *et al.*, 2020] Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020.

[Hao *et al.*, 2021] Qianyue Hao, Fengli Xu, Lin Chen, Pan Hui, and Yong Li. Hierarchical reinforcement learning for scarce medical resource allocation with imperfect information. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2955–2963, 2021.

[Hasanbeig *et al.*, 2018] Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099*, 2018.

[Hasanbeig *et al.*, 2020] Mohammadhosein Hasanbeig, Daniel Kroening, and Alessandro Abate. Deep reinforcement learning with temporal logics. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 1–22. Springer, 2020.

[Heffron and McCauley, 2014] Raphael J Heffron and Darren McCauley. Achieving sustainable supply chains through energy justice. *Applied Energy*, 123:435–437, 2014.

[Huang *et al.*, 2022] Zijian Huang, Meng-Fen Chiang, and Wang-Chien Lee. Line: Logical query reasoning over hierarchical knowledge graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 615–625, 2022.

[Igogo, 2022] Tsisilile Igogo. America's strategy to secure the supply chain for a robust clean energy transition. Technical report, USDOE Office of Policy, Washington DC (United States), 2022.

[Johnson and Smilowitz, 2007] Michael P Johnson and Karen Smilowitz. Community-based operations research. In *OR tools and applications: Glimpses of future technologies*, pages 102–123. INFORMS, 2007.

[Kaufmann, 2018] Emilie Kaufmann. On bayesian index policies for sequential resource allocation. *The Annals of Statistics*, 46(2):842–865, 2018.

[Kronqvist *et al.*, 2021] Jan Kronqvist, Ruth Misener, and Calvin Tsay. Between steps: Intermediate relaxations between big-m and convex hull formulations. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 299–314. Springer, 2021.

[Li and Srikumar, 2019] Tao Li and Vivek Srikumar. Augmenting neural networks with first-order logic. *arXiv preprint arXiv:1906.06298*, 2019.

[Li *et al.*, 2023] Yikuan Li, Chengsheng Mao, Kaixuan Huang, Hanyin Wang, Zheng Yu, Mengdi Wang, and Yuan Luo. Deep reinforcement learning for efficient and

fair allocation of health care resources. *arXiv preprint arXiv:2309.08560*, 2023.

[Lien *et al.*, 2014] Robert W Lien, Seyed MR Iravani, and Karen R Smilowitz. Sequential resource allocation for nonprofit operations. *Operations Research*, 62(2):301–317, 2014.

[Liu *et al.*, 2021] Yongshuai Liu, Jiaxin Ding, Zhi-Li Zhang, and Xin Liu. Clara: A constrained reinforcement learning based resource allocation framework for network slicing. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 1427–1437. IEEE, 2021.

[Malenica *et al.*, 2024] Ivana Malenica, Jeremy R Coyle, Mark J van der Laan, and Maya L Petersen. Adaptive sequential surveillance with network and temporal dependence. *Biometrics*, 80(1):ujad007, 2024.

[Maslen and Paine, 2024] Hannah Maslen and Colin Paine. Ethical resource allocation in policing: Why policing requires a different approach from healthcare. *Criminal Justice Ethics*, 43(1):1–36, 2024.

[Nandwani *et al.*, 2019] Yatin Nandwani, Abhishek Pathak, and Parag Singla. A primal dual formulation for deep learning with constraints. *Advances in neural information processing systems*, 32, 2019.

[Peng *et al.*, 2019] Zedong Peng, Yi Zhang, Yiping Feng, Tuchao Zhang, Zhengguang Wu, and Hongye Su. Deep reinforcement learning approach for capacitated supply chain optimization under demand uncertainty. In *2019 Chinese automation congress (CAC)*, pages 3512–3517. IEEE, 2019.

[Prasath *et al.*, 2024] S Ram Prasath, M Selvi, R Santhana Krishnan, Ezhil E Nithila, P Stella Rose Malar, and G Ram Sankar. Combining fuzzy logic and reinforcement learning for resource management in edge computing. In *2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 1022–1027. IEEE, 2024.

[Pu, 2021] Lida Pu. Fairness of the distribution of public medical and health resources. *Frontiers in public health*, 9:768728, 2021.

[Qin *et al.*, 2021] Zengyi Qin, Yuxiao Chen, and Chuchu Fan. Density constrained reinforcement learning. In *International Conference on Machine Learning*, pages 8682–8692. PMLR, 2021.

[Rantzer, 2001] Anders Rantzer. A dual to lyapunov's stability theorem. *Systems & Control Letters*, 42(3):161–168, 2001.

[Ray *et al.*, 2019] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking Safe Exploration in Deep Reinforcement Learning. 2019.

[Ren *et al.*, 2020] Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. *arXiv preprint arXiv:2002.05969*, 2020.

[Syed *et al.*, 2008] Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, pages 1032–1039, 2008.

[Taylor *et al.*, 2023] Lee-Anne Taylor, Jodee Reid, and Anita Jagroop-Dearing. The impacts of the covid-19 traffic light system on staff in tertiary education in new zealand. *Education Sciences*, 14(1):48, 2023.

[Tessler *et al.*, 2018] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

[Trespalacios and Grossmann, 2015] Francisco Trespalacios and Ignacio E Grossmann. Improved big-m reformulation for generalized disjunctive programs. *Computers & Chemical Engineering*, 76:98–103, 2015.

[Türkay and Grossmann, 1996] Metin Türkay and Ignacio E Grossmann. Disjunctive programming techniques for the optimization of process systems with discontinuous investment costs- multiple size regions. *Industrial & engineering chemistry research*, 35(8):2611–2623, 1996.

[Waga *et al.*, 2022] Masaki Waga, Ezequiel Castellano, Sasinee Pruekprasert, Stefan Klikovits, Toru Takisaka, and Ichiro Hasuo. Dynamic shielding for reinforcement learning in black-box environments, 2022.

[Wu *et al.*, 2024] Zifan Wu, Bo Tang, Qian Lin, Chao Yu, Shangqin Mao, Qianlong Xie, Xingxing Wang, and Dong Wang. Off-policy primal-dual safe reinforcement learning. *arXiv preprint arXiv:2401.14758*, 2024.

[Yang *et al.*, 2020] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.

[Zhang *et al.*, 2023] Libo Zhang, Yang Chen, Toru Takisaka, Bakh Khoussainov, Michael Witbrock, and Jiamou Liu. Learning density-based correlated equilibria for markov games. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 652–660, 2023.

[Zhao *et al.*, 2020] Xiaofan Zhao, Liang Wu, and Ye Qi. The energy injustice of hydropower: Development, resettlement, and social exclusion at the hongjiang and wanmipo hydropower stations in china. *Energy Research & Social Science*, 62:101366, 2020.