Minimizing Negative Side Effects in Cooperative Multi-Agent Systems Using Distributed Coordination

Moumita Choudhury
University of Massachusetts Amherst
Amherst, United States
amchoudhury@cs.umass.edu

Sandhya Saisubramanian
Oregon State University
Corvallis, United States
sandhya.sai@oregonstate.edu

Hao Zhang
University of Massachusetts Amherst
Amherst, United States
hao.zhang@umass.edu

Shlomo Zilberstein
University of Massachusetts Amherst
Amherst, United States
shlomo@cs.umass.edu

ABSTRACT

Autonomous agents engineered for operating in real-world environments frequently encounter undesirable outcomes or Negative Side Effects (NSEs) when working collaboratively alongside other agents. Even when agents can execute their primary assigned tasks optimally when operating in isolation, their training may not account for potential negative interactions that arise in the presence of other agents. We frame the challenge of minimizing NSEs as a Lexicographic Decentralized Markov Decision Process. In this framework, we assume independence of rewards and transitions with respect to the primary assigned tasks while recognizing that addressing negative side effects creates a form of dependence within this context. Furthermore, we focus on minimizing NSEs arising from interactions between a limited subset of agents in the system. We present a lexicographic Q-learning approach to mitigate the NSEs using human feedback models while maintaining near-optimality with respect to the assigned tasks (up to some given slack). Our empirical evaluation across two domains demonstrates that our collaborative approach effectively mitigates NSEs, outperforming non-collaborative methods.

KEYWORDS

AI Safety, Negative Side Effects, Cooperative Multi-agent Systems, Distributed Constraint Optimization Problems

ACM Reference Format:
Moumita Choudhury, Sandhya Saisubramanian, Hao Zhang, and Shlomo Zilberstein. 2024. Minimizing Negative Side Effects in Cooperative Multi-Agent Systems Using Distributed Coordination. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

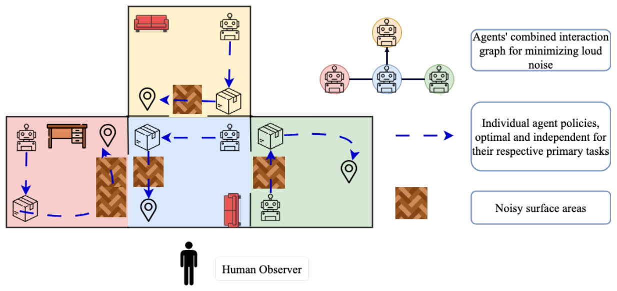

Agents' combined interaction graph for minimizing loud noise
Individual agent policies, optimal and independent for their respective primary tasks
Noisy surface areas
Human Observer

Figure 1: Illustration of multi-agent NSEs in a boxpushing domain. Four boxpushing robots are assigned to push boxes in a large space. The primary policies for pushing boxes do not require coordination. However, certain noisy surfaces in each room produce loud sounds when agents from adjacent rooms simultaneously push boxes across these surfaces. Thus, the noise creates local negative interactions or NSEs among neighboring agents.

1 INTRODUCTION

Autonomous agents operating in the real world frequently generate undesired outcomes [10, 12, 29] that are challenging to rectify during their training phase. Prior works have identified several categories of side effects, i.e., misspecification of rewards in Reinforcement Learning (RL) or goals in symbolic planning [2, 17, 19], distributional shift in the test environment from the training environment [16], reward gaming [6], etc. Lately, there has been a growing focus on scenarios in which an agent's actions directly influence the performance of other entities within the environment [1, 12]. Given the numerous settings where cooperative agents coexist and collaborate [7, 11], it becomes essential to investigate the occurrence of side effects in such multi-agent environments. In this work, we specifically focus on cooperative multi-agent settings where a collective of agents operate in a shared environment, as seen in scenarios like teams of robots within factory warehouses or fleets of autonomous vehicles, but their joint actions leave impacts on the agency of other agents or the environment itself.

In this work, we consider cooperative multi-agent settings where multiple agents are working to complete their assigned tasks. For example, Figure 1 presents a modified boxpushing scenario [19] where some warehouse agents are working collaboratively to push boxes from one place to another. Each agent's individual model provides precise information regarding their rewards and transition dynamics, ensuring optimal performance in their respective primary tasks. However, these models exhibit limitations in accounting for secondary effects stemming from joint actions, such as noise generation when concurrently pushing boxes across specific surfaces characterized by high noise levels. We focus on settings where executing each agent's primary policy in isolation does not result in Negative Side Effects (NSEs), but their combined policy does. Moreover, the agents may initially lack knowledge about the occurrence of NSEs.

Most of the prior works have focused on mitigating NSEs in a single agent setting and mitigated it by i) recomputing the agent's primary reward function [10], ii) incorporating user feedback [29], and iii) incorporating a lexicographic multi-objective approach [19]. Recently, approaches have been developed to model the well-being of other agents by considering the future welfare of the same agents [12] or to facilitate the return of other agents [1, 24]. However, these approaches do not fully scale to multi-agent settings, as only one agent is responsible for taking care of the considerations of other agents. On the contrary, we focus on settings where all the agents are responsible for and expected to cooperate in minimizing the side effects. We adopt a lexicographic multi-objective approach by formulating the problem with a Lexicographic Markov Decision Process (LMDP) due to several reasons: i) it allows us to prioritize the primary objective and guarantee the near-optimal performance of the primary task, ii) it naturally handles different objectives with different and incomparable reward units, and iii) it eliminates time–consuming parameter tuning associated with scalarized objectives, which do not scale well to multi-agent settings [18]. As the minimization of NSEs in this scenario necessitates coordination among the agents, the ongoing challenge lies in identifying an appropriate coordination strategy.

Decentralized Markov Decision Processes (DEC-MDPs) provide an important framework for modeling cooperative multi-agent coordination problems. However, scalability becomes a major concern when it comes to solving general DEC-MDPs. Therefore, researchers have focused on solving restricted versions of DEC-MDPs that are scalable yet rich enough to solve a wide array of practical problems [3, 14]. In our work, we focus on DEC-MDPs with transition independence and locality of interaction, where a network of agents is formed based on each agent's limited interactions with a small number of neighbors. For example, in Figure 1, agents from adjacent rooms establish local interaction networks because their joint actions are causing NSEs in the form of noise. Distributed Constraint Optimization Problems (DCOPs) have been developed over the past two decades to exploit these local interactions [8, 9]. To benefit from both the local interaction structure from DCOP and planning under uncertainty, Network Distributed Partially Observable MDPs (ND-POMDPs) have been developed to solve practical, real-world problems such as sensor networks or coordination of UAVs [14, 27]. Several variations of the Coordinated Q-learning (CQL) approach have been proposed to solve ND-POMDPs because of its capability of solving the problem in a model-free scalable way with only local observability and interactions among neighboring agents [27, 28]. Our particular focus lies in the collective reward structure associated with NSEs, wherein only a subset of agents jointly contribute to the creation of side effects with full local observability among neighboring agents.

We present a combined approach integrating lexicographic multi-objective learning and coordinated Q-Learning to minimize the impacts of NSEs in a multi-agent environment. To the best of our knowledge, there is no off-the-shelf solver for multi-agent lexicographic multi-objective problems. The occurrences of side effects are learned from human feedback, as the agents were initially unaware of the penalties associated with the NSEs. Agents can compute and reuse their primary objective independently. Whenever there is a negative penalty, agents coordinate with their neighbors to form an interaction graph to facilitate coordinated learning. The model-free framework in the coordination process allows the agents to function autonomously without the necessity of sharing model information, thereby safeguarding privacy. Our primary contributions can be summarized as follows: i) formalizing the problem of multi-agent NSE as a lexicographic DEC-MDP with local interaction, ii) defining a way to collect and generalize the joint penalty function from human feedback, iii) presenting a solution approach for minimizing NSEs with a Coordinated Lexicographic Q-learning (C-LQL) solver, and iv) evaluating the performance of our approach and comparing it to non-coordinated and single-agent lexicographic Q-learning approaches.

2 PRELIMINARIES

In this section, we discuss the background necessary to understand our proposed method. Specifically, we provide relevant definitions for Decentralized Markov Decision Processes, Lexicographic Markov Decision Processes, and Distributed Constraint Optimization Problems, which are central to formulating the problem and implementing the proposed approach.

2.1 Decentralized Markov Decision Processes

A Decentralized Markov Decision Process (DEC-MDP) [3, 5] is a multi-agent MDP with n agents which can be defined as a tuple $M' = \langle S', A', P', R' \rangle$ where:

- S' is a finite set of states with an initial state s_0;
- $A' = A_1 \times A_2 \times \ldots \times A_n$ is a finite set of joint actions where A_i is the set of actions for agent i;
- $P' : S' \times A' \times S' \to [0,1]$ denotes the transition function where $P'(s'|(s, < a_1, a_2, \ldots, a_n >)$ is the probability of reaching state s' when executing the joint action $\langle a_1, \ldots, a_n \rangle$ in state s; and
- $R' : S' \times A' \times S' \to \mathbb{R}$ denotes the reward function associated with the agents' assigned task.

Due to the intractability of optimally solving general DEC-MDPs, various works have focused on restricted versions of DEC-MDPs that are easier to solve yet rich enough to represent many practical applications. Examples include Transition and Reward Independent DEC-MDPs, Network Distributed Partially Observable MDPs(ND-POMDPs) [14, 25] which exploits the locality of interactions. We consider the agents to be *reward and transition independent* [3]. A DEC-MDP is transition independent if the actions taken by one agent do not impact another agent's transition or reward for its assigned task. To be specific, M' is transition independent if there exist P_0 through P_n such that

$$P'((s'_0, \ldots, s'_n)|(s_0, \ldots, s_n), (a_0, \ldots, a_n)) = \prod_{i=0}^{n} P_i(.)$$

and is reward independent when

$$R'((s_0,\dots,s_n),(a_0,\dots,a_n),(s'_0,\dots,s'_n)) = \sum_{i=0}^{n} R_i(s_i,a_i,s'_i)$$

As a result, the DEC-MDP model can be solved as n single agent MDPs, with respect to the primary objective [4], and $\pi' = \{\pi_1,\dots,\pi_n\}$, where π_i denotes the primary policy of agent i. A primary policy, π_i, is an optimal policy for M, optimizing the i^{th}agent's primary objective defined by R_i.

2.2 Lexicographic MDP

A lexicographic Markov Decision Process (LMDP) [26] is a multi-objective MDP with lexicographic preferences over the reward functions. An LMDP can be defined as a tuple $M = \langle S, A, T, \mathbf{R}, o, \mathbf{\Delta}\rangle$ where,

- S denotes a finite set of states;
- A denotes a finite set of actions;
- $T : S \times A \times S \rightarrow [0, 1]$ is the transition function;
- $\mathbf{R} = \{R_1,\dots,R_k\}$ is the vector of k reward functions;
- o denotes the preference ordering among the k objectives;
- $\mathbf{\Delta} = \{\delta_1,\dots\delta_k\}$ is the vector of slack variables with each $\delta_i \geq 0$. The slack δ_i denotes the allowed deviation from objective o_i to improve the quality of objective lower priority objective o_j s.t., $j > i$.

The set of value functions is represented as $\mathbf{V} = \{V_1,\dots,V_k\}$ where V_i denotes the value function corresponding to objective o_i, and calculated as:

$$\mathbf{V}^{\pi}(s) = \mathbf{R}(s,\pi(s)) + \gamma \sum_{s'\in S} T(s,\pi(s),s')\mathbf{V}^{\pi}(s'), \forall s \in S.$$

2.3 Distributed Constraint Optimization Problems

A *Distributed Constraint Optimization Problem* (DCOP) can be defined as a tuple $\langle \zeta, X, D, F, \alpha\rangle$ [13] where,

- ζ is a set of agents $\{\zeta_1,\zeta_2,\dots,\zeta_n\}$;
- X is a set of discrete variables $\{x_1,x_2,\dots,x_m\}$, where each variable x_j is controlled by one of the agents $\zeta_i \in \zeta$;
- D is a set of discrete domains $\{D_1,D_2,\dots,D_m\}$, where each D_i corresponds to the domain of variable x_i;
- F is a set of cost functions $\{f_1,f_2,\dots,f_l\}$, where each $f_i \in F$ is defined over a subset $x^i = \{x_{i_1}, x_{i_2}, \dots, x_{i_k}\}$ of variables X, called the *scope* of the function, and the cost for the function f_i is defined for every possible value assignment of x^i, that is, f_i: $D_{i_1} \times D_{i_2} \times \dots \times D_{i_k} \rightarrow \mathbb{R}$, where the arity of the function f_i is k;
- $\alpha : X \rightarrow \zeta$ is a variable-to-agent mapping function that assigns the control of each variable $x_j \in X$ to an agent $\zeta_i \in \zeta$. Each agent can hold several variables. However, in this paper, we assume each agent controls only one variable and use the notation interchangeably.

An optimal solution of a DCOP is an assignment X^* that minimizes the sum of cost functions as shown below[1]:

$$X^* = \underset{X}{\operatorname{argmin}} \sum_{f_i\in F} f_i(x^i).$$

3 PROBLEM FORMULATION

Consider a cooperative multi-agent setting with n agents operating independently to complete their respective assigned tasks, which are their primary objectives, $O_1 = \{o_1,\dots,o_n\}$. The agents operate based on a transition and reward independent DEC-MDP, M' that contains all the necessary information to complete their assigned tasks. However, the agents' models do not fully capture all the objectives in the complex real-world environment in which the agents operate. In this case, there is an additional secondary objective, O_2, that the agents need to optimize to minimize NSEs. The two objectives in M' are: primary assigned tasks (O_1) and mitigating side effects (O_2), where $O_1 > O_2$. Although the agents are transition and reward independent w.r.t. O_1, NSEs occur primarily because of their joint interaction. Such undesirable outcomes arise because the agents compute policies independent of other agents. Furthermore, the agents' model may not account for superfluous details, such as the cumulative effects of joint operation on the environment, that are unrelated to the agents' task.

We make the following assumptions: i) executing the primary policy of each agent in isolation produces no negative side effects, but their joint policy $\pi' = \{\pi_1,\dots\pi_n\}$ produces NSEs, unknown to the agents apriori, ii) the subset of agents interacting with each other to produce NSEs is much smaller than the total number of agents, iii) side effects are undesirable but not catastrophic, and iv) side effects are experienced immediately after joint execution in a state. Building on this, we define multi-agent Negative Side Effects (MANSE), which indicates that the occurrence and penalty for NSE, denoted by R_N, depends on what actions agents perform jointly in a state. We assume a given interaction graph to facilitate the coordination between the agents.

Definition 3.1. Let $G = (X, E)$ be an interaction graph where each node $x_i \in X$ represents an agent i and each hyperlink $l \in E$ connects a subset of agents to form the reward component R_l. $F = \{f_1,\dots,f_l\}$ denotes set the cost functions where f_l represents the cost function associated with each hyperlink l. Moreover, we define $\mathcal{F}_i$ to be the set of functions denoting which function nodes are connected to variable x_i, representing agent i. This hypergraph is formed to facilitate the interaction between agents to optimize the joint penalty where each hyperlink represents a subgroup of agents creating NSEs.

Definition 3.2. The joint penalty function, $R_N : S \times A \rightarrow \mathbf{R}$ for MANSE is a divisible penalty function among subgroups of agents and can be expressed as $R_N(s,a) = \sum_l R_l(s_l,a_l)$ where $l = \{i_1,\dots,i_k\}$ denotes a subgroup of size k. Moreover, $s_l = \langle s_{l_1},\dots,s_{l_k}\rangle$ denotes the state of group l and $a_l = \langle a_{l_1},\dots,a_{l_k}\rangle$ denotes the action of group l.

Solving MANSE involves addressing the following challenges. First, since the NSE penalty is defined over the joint actions, it *does*

[1]For a maximization problem, the argmin operator should be replaced by the argmax operator.

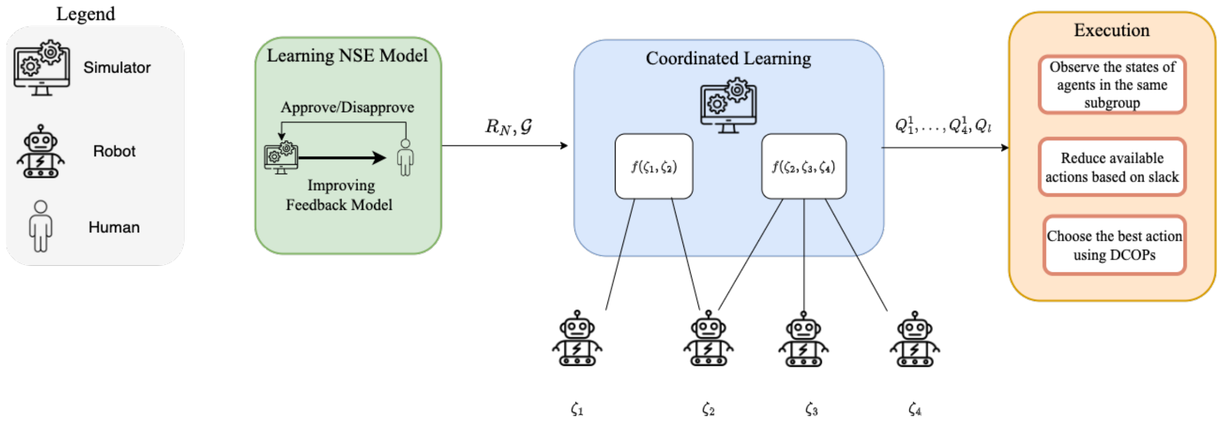


Figure 2: Our framework for minimizing NSEs. $\zeta_1, \ldots, \zeta_4$ denotes the agents, R_N is the learned NSE penalty function from the human feedback model and $\mathcal{G}$ denotes the interaction graph. $f(\ldots)$ denotes the cost associated with each hyperlink, $Q_1^1, \ldots, Q_4^1$ denote the Q values for the primary objective and Q_l denotes the Q value for NSEs caused by group l.

not follow reward independence. Hence, the problem can no longer be solved as n-single agent MDPs. Second, the agents do not have prior knowledge of other agent's models or NSEs. Hence, R_N is unknown initially. Learning to optimize for R_N might deviate the agents from their optimal policies for the primary objective. The maximum allowed deviation for each agent from the optimal objective value corresponding to its primary objective is bounded by its slack value denoted by δ_i for each agent i. In order to determine if the updated policy that minimizes NSEs violates the slack constraints of the agent, we calculate the corresponding interference to its primary objective.

Definition 3.3. The augmented MDP for the given MANSE problem is a lexicographic DEC-MDP (LDEC-MDP), denoted as $\tilde{M} = \langle \tilde{S}, \tilde{A}, \tilde{P}, \tilde{\boldsymbol{R}}, o, \tilde{\boldsymbol{\Delta}} \rangle$. $\tilde{M}$ is a DEC-MDP with two reward functions $\tilde{R} = \{R_1, R_N\}$ where R_1 is the independent reward associated with the primary objective and R_N is the joint reward associated with NSE of joint actions. R_N follows the decomposition described in Definition 3.2. Moreover, $O = \{O_1, O_2\}$ the ordering of the objectives where $O_1 = \{o_1, \ldots, o_n\}$ is the primary objectives associated with the agents' independent assigned tasks described by reward R_1. Here, o_i represents the primary objective for agent i. O_2 denotes the objective to minimize NSEs and $O_1 \succ O_2$. $\tilde{\boldsymbol{\Delta}}$ refers to the collection of Δ for each agent.

Ideally, we would like to solve the LDEC-MDP to reduce negative side effects in the multi-agent setting. In the following section, we propose a method to estimate R_N using various feedback mechanisms and solve the LDEC-MDP using lexicographic Q-learning with DCOPs.

4 FRAMEWORK FOR MINIMIZING NEGATIVE SIDE EFFECTS

In the coordinated learning approach, we assume the existence of a simulator to facilitate the learning. The NSE penalty signals ideally come from a human providing feedback on agents' actions. In this framework, we assume the simulator learns a predictive model of NSEs by gathering information from humans to mimic human feedback. Based on the learned model, the agents form a combined LDEC-MDP to solve. We follow a model-free Lexicographic Q-learning approach [22] using DCOPs to find a lexicographically optimized policy for our problem. Our complete solution framework for minimizing NSEs is illustrated in Figure 2 and involves the following two steps: (1) gathering information about NSEs using human feedback and generalizing them to unseen situations; (2) using a coordinated learning approach to solve the augmented LDEC-MDP.

4.1 Learning Joint NSE Model

During the coordinated learning phase, the oracle, typically representing human feedback, provides signals about undesirable actions. Alternatively, the simulator can simulate the reward signals for the occurrences of NSEs by learning the human feedback model even when perfect human feedback is unavailable. We consider human feedback in the form of approval using two different methods: random queries and trajectories.

Approval based on random queries The Simulator randomly selects joint state-action pairs, without replacement, to query the human, given a budget. The human, in turn, either approves or disapproves the joint action in those states, indicating the NSE occurrence. The approved actions are mapped to zero penalty, $R_N((\vec{s}, \vec{a}) = 0)$. All disapproved actions are mapped to a positive penalty $R_N((\vec{s}, \vec{a}) = k)$ where k is problem-specific. As the samples generated for querying by this approach are *i.i.d.*, it does not introduce any sampling bias in the learning process.

However, in large problems, this approach may require a large number of samples to generate a good estimate of R_N. Given the joint state-action space increases exponentially with the increase in the number of agents, it is often unrealistic to expect a large number of query responses from the human. Hence, we present an alternate querying method where the human provides feedback based on agent trajectories.

Approval based on trajectories In this approach, the simulator presents agent trajectories to the human for feedback, $\vec{T} = (T_1, \ldots, T_n)$, where T_i is the trajectory for i^{th} agent. The trajectories are generated using an ϵ-greedy policy of its optimal primary policy. The human provides feedback by approving or disapproving the actions observed in the trajectories, similar to the *random querying* approach. This approach provides a sample efficient way to gather information about side effects since the feedback is collected for states that are visited by the agents. This approach, however, suffers from bias induced by correlated samples since the states visited are not *i.i.d.*.

Model learning After collecting information about R_N, the Simulator generalizes the observations to unseen situations by training a random forest classifier (RC). The RC model is used to estimate a penalty by predicting the severity of the NSE. Finally, the simulator obtains an NSE penalty function R_N that maps the joint state-action pairs to the NSE penalty value according to their severity. In practice, any classifier may be used to predict NSE occurrence. We assume the collected human feedback is perfect and noise-free, and the generalized feedback data that the simulator uses captures the human feedback model correctly.

4.2 Coordinated Lexicographic Q-learning

In this section, we demonstrate how the agents learn to minimize NSEs jointly with the penalty they receive from the simulator using Coordinated Lexicographic Q-learning (C-LQL). We use a combination of approaches: (1) a lexicographic Q-learning solver for LMDP [22], and (2) a Coordinated Q-learning (CQL) approach that uses a DCOP solver to acquire joint Q-values for NSE minimization [27]. We use a model-free LMDP solver as the backbone of our approach. Such lexicographic solvers work by restricting actions available for each objective according to their priority. Let $A^j_{s,i}$ be the set of available actions that the i^{th} agent has for optimizing o_j in state s and Q^1_i be the set of Q values for optimizing the primary objective of agent i. η^j_i is the state level slack computed from the global slack Δ^j_i for o_j of agent i. We can use the following equations:

$$A^1_{s,i} = A^1_i \tag{1}$$

$$A^2_{s,i} = \{a \in A^1_i | Q^1_i(s,a) \geq \max_{a' \in A^1_{s,i}} Q^1_i(s,a') - \eta^j_i\} \tag{2}$$

In our case, the primary objective is the agents' assigned tasks, O_1, which can be solved by single-agent Q-learning, and therefore, the agents do not coordinate for updating Q_1. Furthermore, at each step of the Q-learning, each agent i shares its pruned action set, A^1_i as the domain, D_i for the DCOP. Let Q_N be the set of Q values for optimizing objective O_2 and derived from the NSE reward function, R_N. Definition 3.2 allows to decompose the joint Q values, Q_N and enables the agents to calculate it in a distributed manner. Q_N can be decomposed among the agents in the following way where l denotes a subgroup of agents defined in Definition 3.2:

$$Q_N = \sum_{l \in E} Q_l(s_l, a_l) \tag{3}$$

In each subgroup, the agents select a delegate agent to store and update the Q tables for each Q_l. Note that the delegates can be chosen randomly for each group. The update rule for each group l can be written as:

$$Q_l(s_l, a_l) = (1-\alpha) Q_l(s^t_l, a^t_l) + \alpha[r^t_l + \gamma * Q_l(s^{t+1}_l, a^*_l)] \tag{4}$$

Here, a^*_l defines the optimal action for group l. Let a^*_i be the optimal action for agent i at each time step. Equation 4 has all the local components that the agents can calculate locally from their subgroups except the optimal action a^*_l which requires the agents to coordinate. The agents then form a DCOP with the following objective to find joint optimal action a^* so that:

$$a^* = \underset{\boldsymbol{a}}{\operatorname{argmin}} \sum_{f_l \in F} f_l(s, \langle a_{l_1}, \ldots, a_{l_k} \rangle) = \underset{\boldsymbol{a}}{\operatorname{argmin}} \sum_{l} Q_l(s_l, a_l) \tag{5}$$

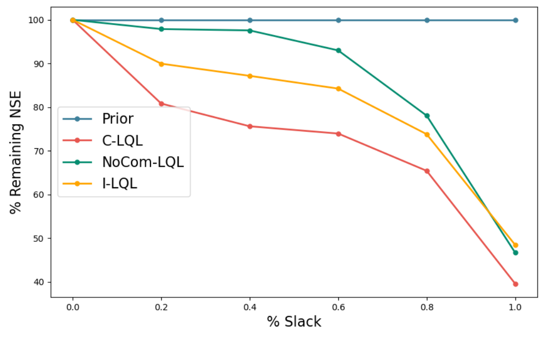
An overview of the C-LQL approach is shown on Algorithm 1. The agents obtain $\langle Q^1_1, \ldots, Q^1_n \rangle$ and Q_N after the end of the learning phase. In the execution phase, the agents first observe the current state of other agents, prune the available action set using Equation 2 and choose the lexicographically optimal action by solving DCOPs. The agents, however, do not require information about other agents' transition or reward models to conduct the optimization. Here we use the Max-sum algorithm [23] for solving the above DCOP because it has less communication overhead compared to other exact solvers.

Algorithm 1: C-LQL for agent i

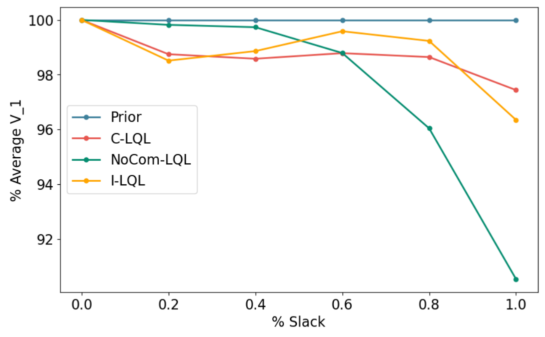
```
Input  : $\tilde{M} = \langle \tilde{S}, \tilde{A}, \tilde{\boldsymbol{R}}, O, \tilde{\Delta}, \gamma \rangle$
1  initialize $Q^1_i$
2  if $i$ is a delegate agent then
3      initialize $Q_l$ for all $l \in \mathcal{F}_i$
4  $s \leftarrow s_0, t \leftarrow 0$
5  while $Q^1_i, Q_l \forall l \in \mathcal{F}_i$ have not converged do
6      $A^2_i \leftarrow$ Prune actions using Equation 2
7      Send $A^1_i$ to each agent $a_j \in l, \forall l \in \mathcal{F}_i$
8      $a^*_i \leftarrow$ Select optimal action using DCOP by optimizing
         Equation 5
9      $s' \leftarrow T(s, a^*_i)$
10     Update $Q^1_i$
11     if $i$ is a delegate agent then
12         Update $Q_l, \forall l \in \mathcal{F}_i$ using Equation 4
13     else
14         Share $(s, a)$ with delegate
15     if $s'$ is terminal then
16         $s \leftarrow s_0$
17     else
18         $s \leftarrow s'$
19     $t \leftarrow t + 1$
```

5 EXPERIMENTAL SETUP

Boxpushing We consider a modified multi-agent boxpushing domain [20, 21] in which the actor is required to minimize the expected time taken to push a box to the goal location. The actions succeed with probability 0.9 and may slide clockwise with probability 0.1. Each state is represented as $\langle x, y, b \rangle$ where x, y denote the agent's location and b is a boolean variable indicating if the agent is pushing the box. We have another unmodeled variable c which indicates the surface type. We assume each agent is working on a designated

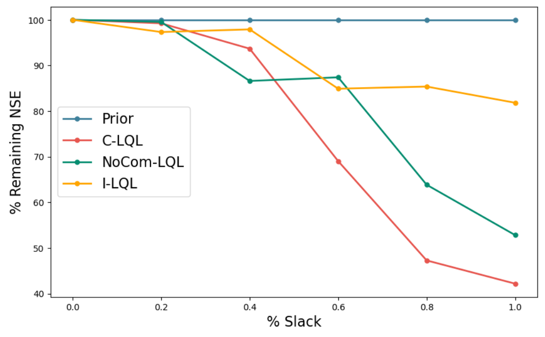


(a) Effect of slack on NSE, 6 agents

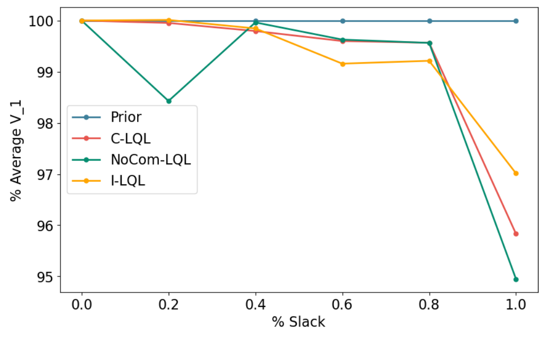


(b) Effect of slack on average primary objective V_1, 6 agents

Figure 3: Effect of slacks on the primary objective and remaining % of NSEs using different approaches for boxpushing problems.



(a) Effect of slack on NSE, 5 agents



(b) Effect of slack on average primary objective V_1, 5 agents

Figure 4: Effect of slacks on the primary objective and remaining % of NSEs using different approaches for painter robot problems.

separate area and so, not violating any of the coordination constraints (i.e., colliding with other agents). Agents create noise when they push the boxes on the surface type $c = N$. When a single agent pushes the box, the noise is negligible. However, when multiple agents push the boxes on surface type $c = N$, it creates side effects. To be precise, pushing the box together on a surface type $c = N$ produces noise and results in NSEs with a penalty of 10, and no NSE otherwise. Although in this example, the side effect penalty is a function of the surface types, in more generalized settings, the side effects can be random and occur in any location of the state space.

Painting Cobots We introduce a painting environment with several narrow corridors where multiple painter robots are deployed in parallel with humans to cover large surface areas more efficiently. We assume each robot has its designated area of the environment to paint. However, there are some narrow corridors that can be inconvenient for humans if blocked or occupied by preferred numbers of agents. The agents are expected to maximize the reward obtained by painting their assigned areas. The robot can move in all four directions and choose to paint (if designated). The primary objective is to finish the painting as quickly as possible. Each state is represented by $\langle x, y, p, c, h \rangle$ where x, y, p denotes the agent's location and the painting areas, c denotes a corridor, and h denotes the existence of a human nearby. When more than one robot enters a corridor at a time, it creates congestion which produces a penalty of 5 if the human is not present in the corridor and a penalty of 10 if the human is in the vicinity.

Baselines We compare two baselines with the Coordinated Lexicographic Q-learning *(C-LQL)* approach. The first is the Independent Lexicographic Q-learning *(I-LQL)* approach [19] which is a model-free modification of the model-based LMDP solver. This approach requires an independent reward model, R_i^1, R_i^2 for each agent i. However, since the reward model R_N for side effects is joint, it is non-trivial to compute the rewards for individual agents. Therefore, we modified the *I-LQL* approach by assuming an oracle individual penalty model. However, such oracle model is hard to derive in practical scenarios because the joint penalty primarily occurs due to the interaction between agents. Our second baseline is the No Communication Lexicographic Q-learning *(NoCom-LQL)* approach where the agents learn individual Q functions by dividing the joint penalty, R_l equally among each member of group l. For the interest of clarity, we make the distinction between *I-LQL* and *(NoCom-LQL)* in the way the individual rewards are obtained. *I-LQL* assumes an individual reward model whereas *NoCom-LQL* receives the actual joint reward by interacting with the environment in a model-free way. Additionally, *Prior* denotes the amount of side effects before minimizing NSEs.

Problem Setup We empirically evaluate our approach with the baselines in two forms of interaction graphs: sparse and dense. The sparse interaction graph is a tree with the maximum degree of 2 and the dense interaction graph is a tree with the maximum degree of 3. We define small problems with the grid size of 5×5 for each agent, totaling the area to $5 \times n * 5$ for n agents in the boxpushing domain. Similarly, a large problem is defined as a larger grid size of 7×7, totaling the area of $7 \times n * 7$.

We use Random forest regression from *sklearn* Python package for our model learning. We follow [26] to determine state-level slack, η_i from a global slack. In all our experiments, we maximize the rewards with $\gamma = 0.95$. The results are averaged over 5 randomly generated instances, each with 5 independent planning process. The values are obtained by 2000 episodes of planning and averaged over 200 runs of execution.

6 RESULTS AND DISCUSSIONS

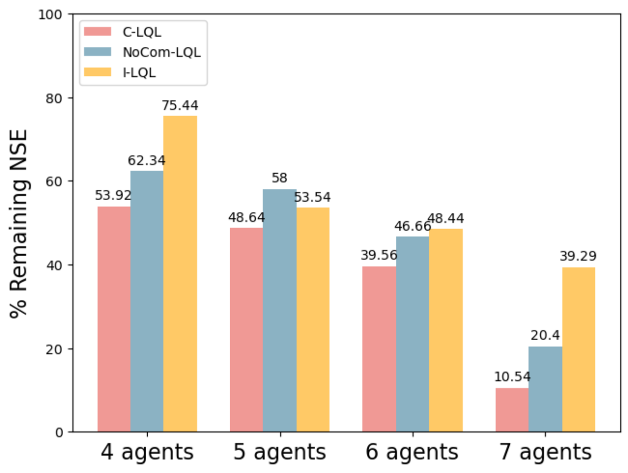


Figure 5: Minimizing Negative Side Effects across different problem sizes in the boxpushing domain

Effect of Slack Determining appropriate slack for MANSE is not trivial. Saisubramanian et al. [19] propose an optimal slack determination algorithm for minimizing single-agent NSEs. However, the slack is optimal only for avoidable NSEs, a special case of NSEs that can be avoided completely while maintaining a path to the goal. In our problem, we do not distinguish between avoidable and unavoidable classes of NSEs because it is nontrivial to determine such classes for multi-agent settings. What is unavoidable for a single-agent setting can be an avoidable setting when multiple agents coordinate. Moreover, it is non-trivial to assign slacks to a group of agents in this case. The lexicographic action restriction proposed in [26] can be conservative for state-level slack allocation [15]. Therefore, we experiment with different variations of slacks with respect to the primary objectives of each agent.

Figure 3 shows the effect of varying slack in 6 agents boxpushing problem. We use a sparse setting with small grids. As the slack

Table 1: Comparison of % of slack used among different agents using different approaches in the box-pushing domain.

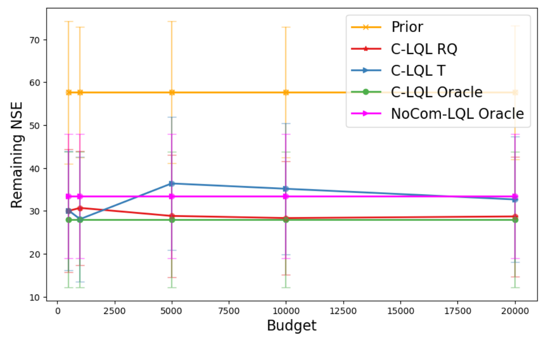
Agents	Approach	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Average
4	Prior	0.0	0.0	0.0	0.0	n/a	n/a	n/a	0.0
	C-LQL	4.60	0.46	7.08	4.09	n/a	n/a	n/a	4.06
	NoCom-LQL	3.50	0.54	0.82	21.64	n/a	n/a	n/a	6.63
	I-LQL	1.32	1.92	0.46	0.21	n/a	n/a	n/a	0.98
5	Prior	0.0	0.0	0.0	0.0	0.0	n/a	n/a	0.0
	C-LQL	8.71	7.71	8.22	5.46	6.17	n/a	n/a	7.25
	NoCom-LQL	0.96	16.98	0.46	9.19	9.67	n/a	n/a	7.45
	I-LQL	3.88	12.88	5.02	2.27	2.12	n/a	n/a	5.23
6	Prior	0.0	0.0	0.0	0.0	0.0	0.0	n/a	0.0
	C-LQL	5.00	3.08	0.67	1.86	3.87	0.91	n/a	2.57
	NoCom-LQL	15.20	7.73	9.09	11.02	5.16	8.62	n/a	9.47
	I-LQL	4.43	2.00	4.51	2.07	1.45	7.46	n/a	3.65
7	Prior	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	C-LQL	27.54	40.97	34.55	45.34	41.87	33.35	27.72	35.91
	NoCom-LQL	37.49	41.81	48.98	51.23	42.43	42.89	54.72	45.65
	I-LQL	28.92	29.58	26.31	12.47	30.75	18.76	37.73	26.36

increases, both the NSEs (Figure 3a) and the average value of the primary objective, V_1 (Figure 3b) tends to decrease. Figure 3b shows the actual slack used by the various Lexicographic Q-learning approaches. Due to the conservative estimation of the local state slacks, the minimum value of the primary objective is still within 90% of the original value. *C-LQL* is able to minimize 60% of the NSEs occurring in the joint state-action space without deviating extensively from the primary objective. *NoCom-LQL* is also able to minimize around 50% of the NSEs. However, it uses more slack than *I-LQL* and *C-LQL* approaches. The coordination among the agents helps to achieve less NSEs without deviating much from the primary objective.

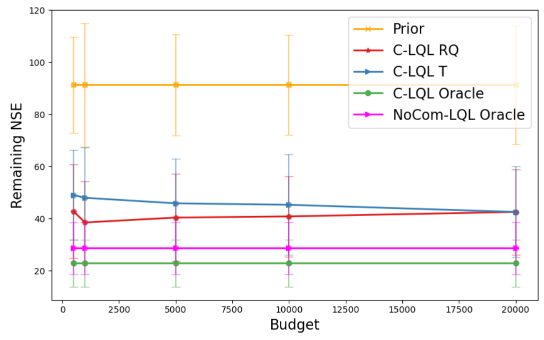
Similarly, Figure 4 shows the effect of varying slack in 5 agents robot painting problems. Notably, *C-LQL* tends to have better performance than *NoCom-LQL* when the given slack is more than 40% of the primary objective (Figure 4a). However, the actual slack used is less than 5% of the primary objective and *C-LQL* requires less slack to produce less NSEs than other approaches (Figure 4b).

Minimizing NSEs We explore the scalability of our approach varying the problem size in number of agents and density. We use sparse and dense problem setups for this experiment. The 4 agent problem is a dense problem with an interaction graph of max degree of 3 where each agent operates on a small grid of size 5×20. The 5, 6, 7 agents problems are sparse problems with a small grid size for 5 and 6 agents problems and a large grid size for the 7 agents problems. As seen from the experiments with slacks, the actual slack used by the agents is much lower than the maximum allowed slack. Therefore, each agent is given a 100% of assigned slack for this experiment. Figure 5 shows the performance of different approaches in different problem sizes. In all the problem setups, *C-LQL* performs better than the other two approaches with the best result of 90% reduction in NSEs in the 7 agent setup. The breakdown of actual slack used by different agents is shown in Table 1. *I-LQL* uses less slack than *C-LQL* and *NoCom-LQL* in most the cases. However, it performs worse in minimizing NSEs than the other two approaches.

Learning Human Feedback Model Figure 6 shows the percentage of remaining NSEs after using *C-LQL* with learned human feedback models with queries. In the boxpushing domain, the random query approach *C-LQL RQ* performs significantly well as the

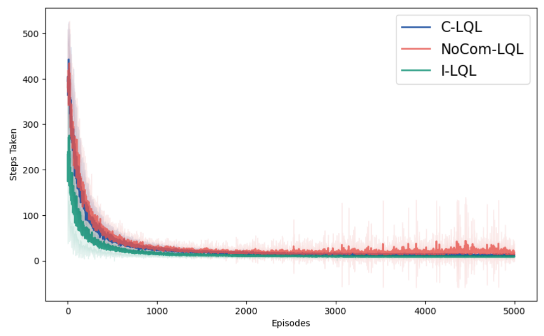


(a) Boxpushing

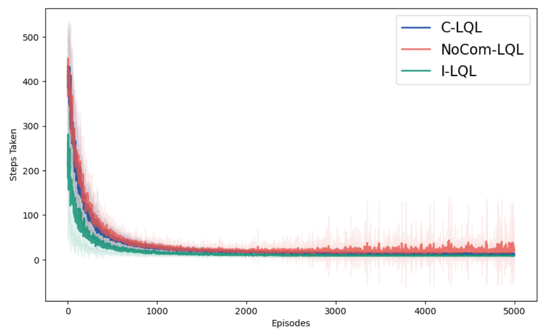


(b) Painter Robots

Figure 6: Effects of learning from human feedback in (a) boxpushing and (b) painter robots domain



(a) 5 agents



(b) 6 agents

Figure 7: Convergence of different approaches in the boxpushing domain

training budget increases (Figure 6a). Trajectory sampling, *C-LQL T*, however, tends to fluctuate with increasing training budgets. This is because it only collects feedback based on trajectories following the $\epsilon-$greedy policy of its optimal primary policy, and therefore, it fails to generalize well to unseen joint state actions. Interestingly, the performance of *C-LQL RQ* is comparable to *C-LQL Oracle* which simulates the problem with a perfect, noise-free human feedback model. In the painter robots domain, initially, the *C-LQL RQ* performs better than the *C-LQL T* approach. With increasing query budget, however, both *C-LQL RQ* and *C-LQL T* performs equally and are able to minimize 50% occurrences of side effects (Figure 6b).

Convergence of Lexicographic Q-learning Figure 7 provides a comparative analysis of convergence between different Lexicographic Q-learning approaches with the increasing number of agents. In both Figure 7a and 7b, *I-LQL* converges faster than both *C-LQL* and *NoCom-LQL*. This is not surprising because *I-LQL* solves a problem with a smaller state space as it treats the problem as a single-agent problem. However, obtaining the perfect single-agent reward model is a non-trivial problem when the NSEs are a factor of agents' joint actions. Furthermore, *C-LQL* and *NoCom-LQL* have an equal convergence rate where both of them converge within 2000 episodes.

7 CONCLUSION AND FUTURE WORK

We formulate the problem of minimizing NSEs in multi-agent systems as a Lexicographic DEC-MDP, which encodes NSEs via a local interaction structure. We propose a framework for minimizing such NSEs using a synergy of approaches from lexicographic multi-objective learning and DCOPs. The proposed model-free lexicographic Q-learning approach facilitates coordination without sharing model information. A key advantage of our approach is that it allows agents to independently optimize their primary objectives while concurrently providing opportunities to learn and adapt to feedback about NSEs during agent deployment. Discovery and handling of NSEs during deployment without compromising the agents' performance with their primary assigned tasks is crucial in many domains. Hence, the proposed approach allows the agents to coordinate without the need to suspend operations and redesign their primary reward function, which may necessitate extensive testing.

Our proposed framework is shown empirically to be effective in minimizing undesirable side effects. Our analysis shows that *C-LQL* minimizes NSEs in different problem settings better than the uncoordinated version, without using much slack. In future work, we aim to extend our approach to handle a more general class of multi-agent problems where the side effects are generated by dynamic interactions among subsets of agents.

8 ACKNOWLEDGMENTS

This work was supported in part by NSF grants 1954782 and 2326054, and by ONR grant N00014-23-1-2171.

REFERENCES

[1] Parand Alizadeh Alamdari, Toryn Q Klassen, Rodrigo Toro Icarte, and Sheila A McIlraith. 2021. Avoiding negative side effects by considering Others. In *Safe and Robust Control of Uncertain Systems Workshop at NeurIPS*.

[2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[3] Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V Goldman. 2003. Transition-independent decentralized markov decision processes. In *Proceedings of the Second International Joint Conference on Autonomous agents and Multiagent systems*. 41–48.

[4] Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V Goldman. 2004. Solving transition independent decentralized markov decision processes. *Journal of Artificial Intelligence Research* 22 (2004), 423–455.

[5] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research* 27, 4 (2002), 819–840.

[6] Jack Clark and Dario Amodei. 2016. Faulty reward functions in the wild. *Internet: https://blog. openai. com/faulty-reward-functions* (2016).

[7] Raffaello D'Andrea. 2012. A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering* 9, 4 (2012), 638–639.

[8] Alessandro Farinelli, Alex Rogers, and Nick R Jennings. 2014. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. *Autonomous Agents and Multi-agent Systems* 28 (2014), 337–380.

[9] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. 2018. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research* 61 (2018), 623–698.

[10] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. 2017. Inverse reward design. *Advances in neural information processing systems* 30 (2017).

[11] Ryan Hoque, Lawrence Yunliang Chen, Satvik Sharma, Karthik Dharmarajan, Brijen Thananjeyan, Pieter Abbeel, and Ken Goldberg. 2023. Fleet-dagger: Interactive robot fleet learning with scalable human supervision. In *Conference on Robot Learning*. PMLR, 368–380.

[12] Victoria Krakovna, Laurent Orseau, Ramana Kumar, Miljan Martic, and Shane Legg. 2019. Penalizing side effects using stepwise relative reachability. In *IJCAI AI Safety Workshop*.

[13] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. 2005. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence* 161, 1-2 (2005), 149–180.

[14] Ranjit Nair, Pradeep Varakantham, Milind Tambe, and Makoto Yokoo. 2005. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *AAAI*, Vol. 5. 133–139.

[15] Luis Enrique Pineda, Kyle Hollins Wray, and Shlomo Zilberstein. 2015. Revisiting multi-objective MDPs with relaxed lexicographic preferences. In *2015 AAAI Fall Symposium Series*.

[16] Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2008. *Dataset shift in machine learning.* Mit Press.

[17] Ramya Ramakrishnan, Ece Kamar, Besmira Nushi, Debadeepta Dey, Julie Shah, and Eric Horvitz. 2019. Overcoming blind spots in the real world: Leveraging complementary abilities for joint execution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6137–6145.

[18] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research* 48 (2013), 67–113.

[19] Sandhya Saisubramanian, Ece Kamar, and Shlomo Zilberstein. 2020. A multi-objective approach to mitigate negative side effects. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Yokohama, Japan, 354–361.

[20] Sandhya Saisubramanian, Ece Kamar, and Shlomo Zilberstein. 2022. Avoiding negative side effects of autonomous systems in the open world. *Journal of Artificial Intelligence Research* 74 (2022), 143–177.

[21] Sven Seuken and Shlomo Zilberstein. 2007. Improved memory-bounded dynamic programming for decentralized POMDPs. *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (2007).

[22] Joar Skalse, Lewis Hammond, Charlie Griffin, and Alessandro Abate. 2022. Lexicographic multi-objective reinforcement learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*.

[23] Ruben Stranders, Alessandro Farinelli, Alex Rogers, and Nick Jennings. 2009. Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*.

[24] Alexander Matt Turner, Dylan Hadfield-Menell, and Prasad Tadepalli. 2020. Conservative agency via attainable utility preservation. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 385–391.

[25] Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. 2014. Decentralized stochastic planning with anonymity in interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28.

[26] Kyle Wray, Shlomo Zilberstein, and Abdel-Illah Mouaddib. 2015. Multi-objective MDPs with conditional lexicographic reward preferences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.

[27] Chongjie Zhang and Victor Lesser. 2011. Coordinated multi-agent reinforcement learning in networked distributed POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 25. 764–770.

[28] Chongjie Zhang and Victor Lesser. 2013. Coordinating multi-agent reinforcement learning with limited communication. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*. 1101–1108.

[29] Shun Zhang, Edmund H Durfee, and Satinder Singh. 2018. Minimax-regret querying on side effects for safe optimality in factored markov decision processes.. In *Proceedings of the Twenty-sixth International Joint Conferences on Artificial Intelligence*. 4867–4873.