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

Moumita Choudhury[1], **Sandhya Saisubramanian**[2], **Hao Zhang**[1], **Shlomo Zilberstein**[1]

[1] College of Information and Computer Sciences, University of Massachusetts Amherst
[2] School of Electrical Engineering and Computer Science, Oregon State University
amchoudhury@umass.edu, sandhya.sai@oregonstate.edu, hao.zhang@umass.edu, shlomo@umass.edu

Abstract

Autonomous agents 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 task 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 which we assume independence of rewards and transitions with respect to the primary assigned tasks, but recognize that addressing negative side effects creates a form of dependence among the agents. 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.

Introduction

Autonomous agents operating in the real world frequently generate undesired outcomes (Hadfield-Menell et al. 2017; Zhang, Durfee, and Singh 2018; Krakovna et al. 2019) that are challenging to rectify during their training phase. Prior works have identified several categories of side effects, such as misspecification of rewards in reinforcement learning (RL) or goals in symbolic planning (Amodei et al. 2016; Saisubramanian, Kamar, and Zilberstein 2020; Ramakrishnan et al. 2019), distributional shift in the deployed environment (Quinonero-Candela et al. 2008), and reward gaming (Clark and Amodei 2016). 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 (Krakovna et al. 2019; Alamdari et al. 2021). Given the numerous settings where cooperative agents coexist and collaborate (D'Andrea 2012; Hoque et al. 2023), it becomes essential to investigate the occurrence of side effects in such multi-agent environments.

This work focuses on cooperative multi-agent settings, such as robot teams in warehouses or fleets of autonomous vehicles. While individual agents may excel in their primary tasks, their joint actions generate unforeseen side effects by leaving impacts on the agency of other agents or the environment itself, illustrated in Fig. 1. We address scenarios where isolated execution of agents' policies is side-effect-free, but their combined actions induce *negative side effects* (NSEs), initially unknown to the agents.

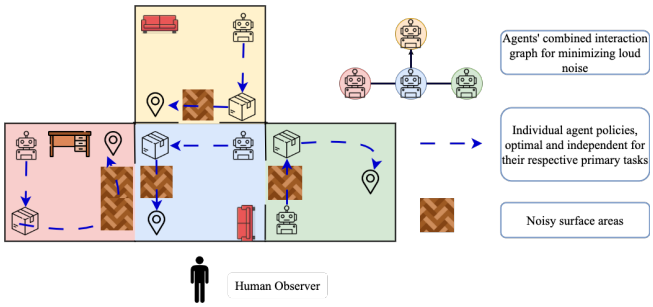


Figure 1: Illustration of multi-agent negative side effects (NSEs) in a boxpushing domain. Four robots collaborate by pushing boxes in a large space. Accomplishing the primary objective of each robot does not require coordination, but the surfaces in each room produce loud noise when adjacent agents push boxes simultaneously.

Most prior works have focused on mitigating NSEs in a single agent setting by (1) recomputing the agent's primary reward function (Hadfield-Menell et al. 2017), (2) incorporating user feedback (Zhang, Durfee, and Singh 2018), and (3) incorporating a lexicographic multi-objective approach (Saisubramanian, Kamar, and Zilberstein 2020). Recently, approaches have been developed to model the well-being of other agents by considering the future welfare of the same agents (Krakovna et al. 2019) or to facilitate the return of other agents (Turner, Hadfield-Menell, and Tadepalli 2020; Alamdari et al. 2021). 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. In contrast, we focus on settings where all the agents are responsible for and expected to cooperate in minimizing the undesired side effects.

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 (Becker et al. 2003; Nair et al. 2005). 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. Distributed constraint optimization problems (DCOPs) have been developed over the past two decades to exploit these local interactions (Farinelli, Rogers, and Jennings 2014; Fioretto, Pontelli, and Yeoh 2018). 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 problems such as coordination of sensor networks or UAVs (Nair et al. 2005; Zhang and Lesser 2011). Several variations of the coordinated Q-learning (CQL) approach have been proposed to solve ND-POMDPs, which can solve the problem in a model-free scalable way with only local observability and interactions among neighboring agents (Zhang and Lesser 2011; 2013). 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 adopt a lexicographic multi-objective approach by formulating the problem with a Lexicographic Markov Decision Process (LMDP) (Wray, Zilberstein, and Mouaddib 2015). 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 existing 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 policy 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 sharing model information, thereby safeguarding privacy. Our primary contributions are fourfold: (1) formalizing the problem of multi-agent NSEs as a lexicographic DEC-MDP with local interaction, (2) defining a way to collect and generalize the joint penalty function from human feedback, (3) presenting a solution approach for minimizing NSEs with a *coordinated lexicographic Q-learning* (C-LQL) solver, and (4) evaluating the performance of our approach and comparing it to non-coordinated and single-agent lexicographic Q-learning approaches.

Problem Formulation

Consider a cooperative multi-agent setting with n agents operating independently to complete their respective assigned tasks, which are their primary objectives, $O_1 = \{o_1, \ldots, o_n\}$. 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 minimize NSEs. The two objectives in M' are: primary assigned tasks (O_1) and mitigating side effects (O_2), where $O_1 \succ O_2$. 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' primary task.

We make the following assumptions: (1) executing the primary policy of each agent in isolation produces no negative side effects, but their joint policy $\pi' = \{\pi_1, \ldots \pi_n\}$ produces NSEs, unknown to the agents apriori, (2) the subset of agents interacting with each other to produce NSEs is much smaller than the total number of agents, (3) side effects are undesirable but not catastrophic, and (4) side effects result immediately from the joint execution in a state. Building on this, we define *multi-agent negative side effects* (MANSE), in which the occurrence and penalty for NSEs, 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 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. $F = \{f_1, \ldots, f_l\}$ 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 2 *The joint penalty function, $R_N : S \times A \rightarrow \boldsymbol{R}$ 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 $l = \{i_1, \ldots, i_k\}$ denotes a subgroup of size k. Moreover, $s_l = \langle s_{l_1}, \ldots, s_{l_k} \rangle$ denotes the state of group l and $a_l = \langle a_{l_1}, \ldots, a_{l_k} \rangle$ denotes the action of group l.*

Solving MANSE presents the following challenges. First, since the NSE penalty is defined over the joint actions, it *does 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 may require agents to deviate from their optimal policies for the primary objective. The maximum allowed deviation for each agent from the optimal primary objective value is bounded by its *slack* value denoted by δ_i for each agent i. To determine if the updated policy that minimizes NSEs violates the slack constraint, we calculate the interference with the corresponding primary objective.

Definition 3 *The augmented MDP for a given MANSE problem is a lexicographic DEC-MDP (LDEC-MDP), which is a multi-agent extension of LMDP (Wray, Zilberstein, and*

Mouaddib 2015), 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 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* $\boldsymbol{\Delta}$ *for each agent.*

In order to reduce negative side effects in a multi-agent setting we have to to solve the corresponding LDEC-MDP. 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.

Framework for Minimizing NSEs

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 (Skalse et al. 2022) using DCOPs to find a lexicographically optimized policy for our problem. Our solution method involves the following two steps: (1) gathering information about NSEs using human feedback and generalizing them to unseen situations, and (2) using a coordinated learning approach to solve the augmented LDEC-MDP.

Learning a Joint NSE Model

During the coordinated learning phase, the oracle, 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 methods: random queries and trajectories.

Approval based on random queries In this approach, 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.

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.

Coordinated Lexicographic Q-learning

We now 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 (Skalse et al. 2022), and (2) a coordinated Q-learning (CQL) approach that uses a DCOP solver to acquire joint Q-values for NSE minimization (Zhang and Lesser 2011). 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 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 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:

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

Algorithm 1: C-LQL for agent i

Input : $\tilde{M} = \langle \tilde{S}, \tilde{A}, \tilde{\boldsymbol{R}}, O, \tilde{\boldsymbol{\Delta}}, \gamma \rangle$

Initialize Q_i^1
if *i is a delegate agent* **then**
  initialize Q_l for all $l \in \mathcal{F}_i$
$s \leftarrow s_0, t \leftarrow 0$
while $Q_i^1, Q_l \forall l \in \mathcal{F}_i$ *have not converged* **do**
  $A_i^2 \leftarrow$ Prune actions using Equation 2
  Send A_i^1 to each agent $a_j \in l, \forall l \in \mathcal{F}_i$
  $a_i^* \leftarrow$ Select optimal action using DCOP by optimizing Equation 5
  $s' \leftarrow T(s, a_i^*)$
  Update Q_i^1
  if *i is a delegate agent* **then**
    Update $Q_l, \forall l \in \mathcal{F}_i$ using Equation 4
  else
    Share (s, a) with delegate
  if *s' is terminal* **then**
    $s \leftarrow s_0$
  else
    $s \leftarrow s'$
  $t \leftarrow t + 1$

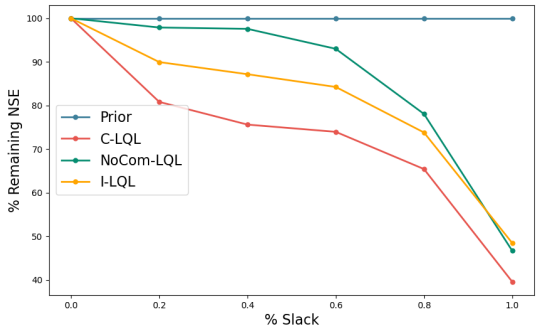
Here, a_l^* defines the optimal action for group l. Let a_i^* be the optimal action for agent i at each time step. Equation 4 has all the local components that the agents can calculate locally from their subgroups except the optimal action a_l^* which requires the agents to coordinate. The agents then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

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

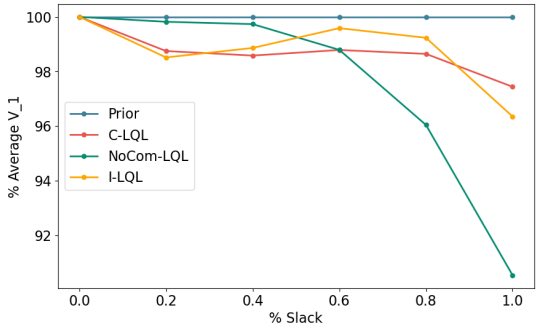
Algorithm 1 summarizes the C-LQL approach. The agents obtain $\langle Q_1^1, \ldots, Q_n^1 \rangle$ and Q_N after the end of the learning phase. In the execution phase, the agents first observe the current state of other agents, prune the available action set using Equation 2 and choose the lexicographically 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 (Stranders et al. 2009) for solving the above DCOP because it has less communication overhead compared to other exact solvers.

Experimental Setup

Boxpushing We consider a modified multi-agent boxpushing domain (Saisubramanian, Kamar, and Zilberstein 2022; Seuken and Zilberstein 2007) 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. Another unmodeled variable, c, indicates the surface type. We assume each agent is working on a designated separate area and so, not violating any of the coordination constraints (i.e., colliding with other agents). 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.



(a) Effect of slack on NSEs w/ 6 agents

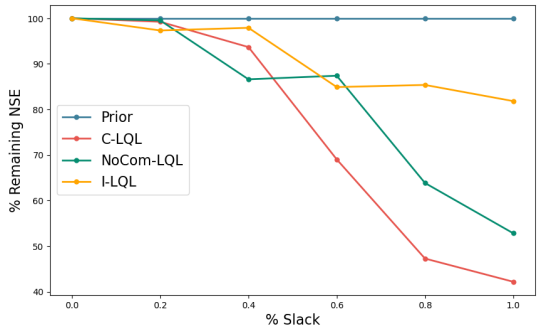


(b) Effect of slack on primary objective V_1 w/ 6 agents

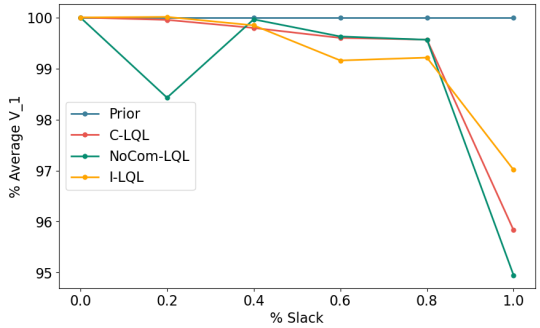
Figure 2: Effect of slack on the primary objective and remaining % of NSEs using different approaches for boxpushing problems.

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 a certain 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. 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 nearby.

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 (Saisubramanian, Kamar, and Zilberstein 2020), which is a model-free variation of the model-based LMDP solver. This approach requires an independent reward model, R_i^1, R_i^2 for each agent i. However, because R_N is a



(a) Effect of slack on NSE w/ 5 agents



(b) Effect of slack on primary objective V_1 w/ 5 agents

Figure 3: Effect of slack on the primary objective and remaining % of NSEs using different approaches for painter robot problems.

joint reward model, 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 practice 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 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. *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 (Wray, Zilberstein, and Mouaddib 2015) to determine state-level slack, η_i from a global slack. In all our experiments, we maximize the rewards with γ=0.95. Results are averaged over 5 randomly generated instances, each with 5 independent planning processes. The values are obtained by 2000 episodes of planning and averaged over 200 runs.

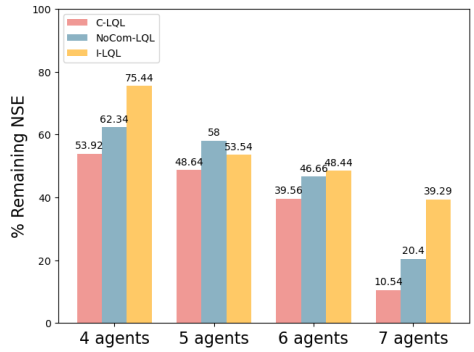


Figure 4: Minimizing negative side effects across different problem sizes in the boxpushing domain.

Results and Discussions

Effect of slack Determining appropriate slack for MANSE is not trivial. Saisubramanian, Kamar, and Zilberstein, 2020 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 nontrivial to assign slacks to a group of agents in this case. The lexicographic action restriction proposed in (Wray, Zilberstein, and Mouaddib 2015) can be conservative for state-level slack allocation (Pineda, Wray, and Zilberstein 2015). Therefore, we experiment with different variations of slacks with respect to the primary objectives of each agent.

Fig. 2 shows the effect of varying slack in 6 agents boxpushing problem. We use a sparse setting with small grids. As the slack increases, both the NSEs (Fig. 2a) and the average value of the primary objective, V_1 (Fig. 2b) tends to decrease. Fig. 2b 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.

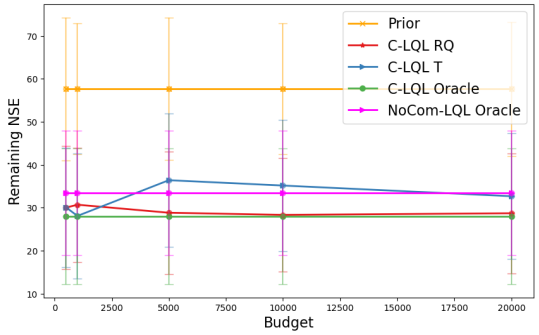
Similarly, Fig. 3 shows the effect of varying slack in 5 agents robot painting problems. Notably, *C-LQL* tends to have better performance than *NoCom-LQL* when the given slack is more than 40% of the primary objective (Fig. 3a). However, the actual slack used is less than 5% of the primary objective and *C-LQL* requires less slack to produce less NSEs than other approaches (Fig. 3b).

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

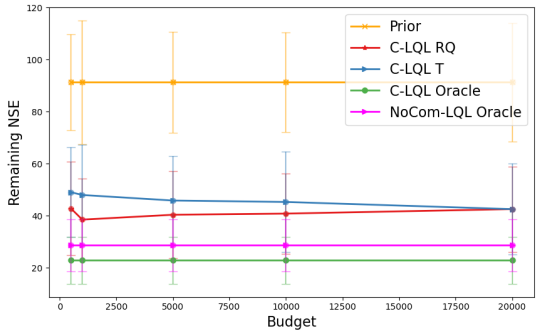
Agents	Approach	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Average
4	Prior	0.0	0.0	0.0	0.0	n/a	n/a	n/a	0.0
	C-LQL	4.60	0.46	7.08	4.09	n/a	n/a	n/a	4.06
	NoCom-LQL	3.50	0.54	0.82	21.64	n/a	n/a	n/a	6.63
	I-LQL	1.32	1.92	0.46	0.21	n/a	n/a	n/a	0.98
5	Prior	0.0	0.0	0.0	0.0	0.0	n/a	n/a	0.0
	C-LQL	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

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. 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. Fig. 4 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 of the cases. However, it performs worse in minimizing NSEs than the other two approaches.

Learning human feedback models Fig. 5 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 better as the training budget increases (Fig. 5a). 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 *C-LQL T*. With increasing query budget, however, both methods perform equally well with a 50% reduction in NSEs (Fig. 5b).



(a) Boxpushing



(b) Painter Robots

Figure 5: Effects of learning from human feedback in 5 agents (a) boxpushing and (b) painter robots domains.

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.

Acknowledgments

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

References

Alamdari, P. A.; Klassen, T. Q.; Icarte, R. T.; and McIlraith, S. A. 2021. Avoiding negative side effects by considering others. In *NeurIPS Workshop on Safe and Robust Control of Uncertain Systems*.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.

Becker, R.; Zilberstein, S.; Lesser, V.; and Goldman, C. V. 2003. Transition-independent decentralized Markov decision processes. In *Proceedings of the Second International Joint Conference on Autonomous agents and Multiagent systems*, 41–48.

Clark, J., and Amodei, D. 2016. Faulty reward functions in the wild. *Internet: https://blog.openai.com/faulty-reward-functions*.

D'Andrea, R. 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):638–639.

Farinelli, A.; Rogers, A.; and Jennings, N. R. 2014. Agent-based decentralised coordination for sensor networks using the Max-Sum algorithm. *Autonomous Agents and Multi-agent Systems* 28:337–380.

Fioretto, F.; Pontelli, E.; and Yeoh, W. 2018. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research* 61:623–698.

Hadfield-Menell, D.; Milli, S.; Abbeel, P.; Russell, S. J.; and Dragan, A. 2017. Inverse reward design. In *Advances in Neural Information Processing Systems*, 6765–6774.

Hoque, R.; Chen, L. Y.; Sharma, S.; Dharmarajan, K.; Thananjeyan, B.; Abbeel, P.; and Goldberg, K. 2023. Fleet-dagger: Interactive robot fleet learning with scalable human supervision. In *Proceedings of the Conference on Robot Learning*, 368–380.

Krakovna, V.; Orseau, L.; Kumar, R.; Martic, M.; and Legg, S. 2019. Penalizing side effects using stepwise relative reachability. In *IJCAI Workshop on AI Safety*.

Nair, R.; Varakantham, P.; Tambe, M.; and Yokoo, M. 2005. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the AAAI Conference on AI*, 133–139.

Pineda, L. E.; Wray, K. H.; and Zilberstein, S. 2015. Revisiting multi-objective MDPs with relaxed lexicographic preferences. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*.

Quinonero-Candela, J.; Sugiyama, M.; Schwaighofer, A.; and Lawrence, N. D. 2008. *Dataset shift in machine learning*. MIT Press.

Ramakrishnan, R.; Kamar, E.; Nushi, B.; Dey, D.; Shah, J.; and Horvitz, E. 2019. Overcoming blind spots in the real world: Leveraging complementary abilities for joint execution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6137–6145.

Saisubramanian, S.; Kamar, E.; and Zilberstein, S. 2020. A multi-objective approach to mitigate negative side effects. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 354–361.

Saisubramanian, S.; Kamar, E.; and Zilberstein, S. 2022. Avoiding negative side effects of autonomous systems in the open world. *Journal of Artificial Intelligence Research* 74:143–177.

Seuken, S., and Zilberstein, S. 2007. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, 344–351.

Skalse, J.; Hammond, L.; Griffin, C.; and Abate, A. 2022. Lexicographic multi-objective reinforcement learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 3430–3436.

Stranders, R.; Farinelli, A.; Rogers, A.; and Jennings, N. 2009. Decentralised coordination of mobile sensors using the Max-Sum algorithm. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 299–304.

Turner, A. M.; Hadfield-Menell, D.; and Tadepalli, P. 2020. Conservative agency via attainable utility preservation. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 385–391.

Wray, K.; Zilberstein, S.; and Mouaddib, A.-I. 2015. Multi-objective MDPs with conditional lexicographic reward preferences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 3418–3424.

Zhang, C., and Lesser, V. 2011. Coordinated multi-agent reinforcement learning in networked distributed POMDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 764–770.

Zhang, C., and Lesser, V. 2013. Coordinating multi-agent reinforcement learning with limited communication. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-agent Systems*, 1101–1108.

Zhang, S.; Durfee, E. H.; and Singh, S. 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.