A Multi-Objective Approach to Mitigate Negative Side Effects

Sandhya Saisubramanian[1] , **Ece Kamar**[2] and **Shlomo Zilberstein**[1]
[1]University of Massachusetts, Amherst, MA
[2]Microsoft Research, Redmond, WA
saisubramanian@cs.umass.edu, eckamar@microsoft.com, shlomo@cs.umass.edu

Abstract

Agents operating in unstructured environments often create *negative side effects* (NSE) that may not be easy to identify at design time. We examine how various forms of human feedback or autonomous exploration can be used to learn a penalty function associated with NSE during system deployment. We formulate the problem of mitigating the impact of NSE as a multi-objective Markov decision process with lexicographic reward preferences and slack. The slack denotes the maximum deviation from an optimal policy with respect to the agent's primary objective allowed in order to mitigate NSE as a secondary objective. Empirical evaluation of our approach shows that the proposed framework can successfully mitigate NSE and that different feedback mechanisms introduce different biases, which influence the identification of NSE.

1 Introduction

Autonomous agents acting in the open world typically operate based on models that are carefully designed and tested with respect to some given *primary* objective, but inevitably details in the environment that are unrelated to the agent's primary objective are ignored [Ramakrishnan *et al.*, 2019]. The incompleteness of any given model is unavoidable due to practical limitations in model specification (the ramification and qualification problems) and due to the limited information that may be available during the design phase [Dietterich, 2017; Saisubramanian *et al.*, 2019]. In this work, we consider an agent operating using a model $\tilde{M}$, which does not fully represent the real world but includes all the details necessary to achieve the agent's primary assigned objective. As a result of the limited fidelity of $\tilde{M}$, the agent's actions may have unmodeled, undesirable *negative side effects* (NSE) in some states [Amodei *et al.*, 2016].

We focus on NSE that are undesirable but not prohibitive. For example, consider an agent that aims to push a box B from location L_1 to L_2 (Figure 1) as quickly as possible. The agent's model $\tilde{M}$ *accurately* represents the reward for pushing the box to L_2, along with the associated transition dynamics, which are essential to achieve its primary objective. But its model may not include other details such as the type of rug and the impact of pushing the box over the rug, as these details were not considered relevant to the task during system design and initial testing. Consequently, when executing the policy computed using $\tilde{M}$, the agent pushes the box over the rug, dirtying the rug as a side effect.

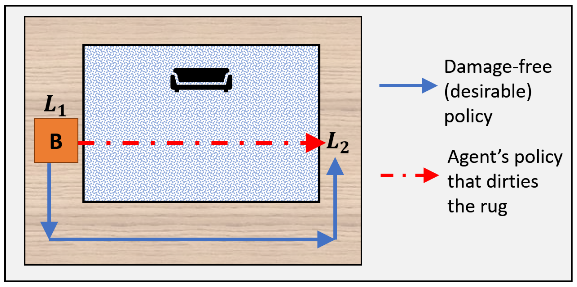

L_1
B
L_2
Damage-free (desirable) policy
Agent's policy that dirties the rug

Figure 1: Illustration of NSE in the boxpushing domain: A policy based on $\tilde{M}$ dirties the rug (shaded area).

Learning to detect and minimize NSE is critical for safe deployment of autonomous systems and to support long-term autonomy in the real-world [Zilberstein, 2015]. One possible approach to avoid NSE is to entirely redesign the agent's model every time NSE are identified. This may corrupt the reward function of the agent's primary objective and hence will likely require suspension of operation until the newly derived policies could be exhaustively evaluated and deemed safe for autonomous operation. Such an approach to avoid NSE is inefficient for complex systems such as autonomous vehicles that operate based on a model that has been carefully designed and tested for critical safety aspects such as yielding to pedestrians and conforming to traffic rules. The key question is how could *deployed* agents respond to feedback about NSE and learn to avoid them.

Existing works (outlined in Table 1) mitigate NSE by recomputing the reward function for the agent's primary objective [Hadfield-Menell *et al.*, 2017], by collecting user feedback about which features in the environment can be altered by the agent [Zhang *et al.*, 2018], or by minimizing deviations from a baseline state [Krakovna *et al.*, 2019]. All these approaches target settings with avoidable NSE and assume that the agent has prior knowledge about the cause and occurrence of NSE. Real-world situations, however, tend to violate

Characteristics of Different Approaches	[Hadfield-Menell *et al.*, 2017]	[Zhang *et al.*, 2018]	[Krakovna *et al.*, 2019]	Our Approach
Bounded guarantees with respect to agent's assigned task	✗	✗	✗	✓
No prior knowledge about NSE occurrence is required	✗	✗	✗	✓
Generalize learned NSE across states	✓	✗	✗	✓
Ability to distinguish and handle different magnitudes of NSE	✗	✓	✓	✓
Multi-objective approach	✗	✗	✓	✓

Table 1: An overview of the characteristics of different approaches in mitigating NSE.

these assumptions. Furthermore, in situations where the primary objective is prioritized, the user may be willing to trade-off some NSE for solution quality. For example, if the agent takes ten times longer to push the box so as to avoid the rug area, the user may be willing to tolerate some dirt on the rug instead. The existing approaches, however, do not guarantee bounded performance with respect to the agent's primary objective when minimizing the side effects.

We propose a multi-objective approach that exploits the reliability of the existing model with respect to the primary objective, while allowing a deployed agent to learn to avoid NSE as much as possible. The agent's model is augmented with a secondary reward function that represents the penalty for NSE of its actions. Hence, the problem is formulated as a multi-objective Markov decision process with lexicographic reward preferences (LMDP) and slack [Wray *et al.*, 2015]. A lexicographic approach is adopted because (1) the reward associated with primary objective is prioritized, (2) the reward for the primary objective and the penalty for NSE may have different units, such as time taken to push a box and the cost of cleaning a rug, and (3) alternative scalarization methods require non-trivial parameter tuning [Roijers *et al.*, 2013].

The agent's primary objective is to achieve its assigned task, while the secondary objective is to minimize NSE. The slack denotes the acceptable deviation from the optimal expected reward of its primary objective, in order to minimize NSE. The agent may not have any initial knowledge about the NSE. The solution framework utilizes a three-step approach to detect and mitigate NSE (Figure 2): (1) first the agent collects data about NSE penalty through a feedback mechanism, (2) the agent learns a predictive model of the penalty for NSE that generalizes the available data, and (3) the agent replans using a model augmented with the learned NSE penalty.

We investigate the efficiency of different feedback approaches to learn about NSE, including oracle feedback and agent exploration. The oracle feedback typically represents some type of human feedback such as penalty signals, action approval, corrections of the agent's behavior, or demonstrations of correct ways to perform the task. In learning by exploration, the agent collects data about NSE by exploring the environment. Our experiments demonstrate the benefits of our approach in mitigating both avoidable and unavoidable NSE. The results also indicate how the sampling biases introduced by the different types of feedback influence the identification of NSE.

Our primary contributions are: (1) formalizing the problem of mitigating NSE as a multi-objective MDP with slack; (2) presenting a solution approach to update the agent's policy by learning about NSE as a secondary reward function and estimating the minimum slack required to avoid NSE; (3) studying various types of feedback mechanisms to learn the penalty associated with side effects; and (4) evaluating the performance and analyzing the bias associated with each feedback mechanism.

2 Preliminaries: Lexicographic MDP

A lexicographic Markov decision process (LMDP) [Wray *et al.*, 2015] is a multi-objective MDP with lexicographic preferences over reward functions. LMDPs are particularly convenient to model multi-objective MDPs with competing objectives and with an inherent lexicographic ordering over them. An LMDP is denoted by the tuple $M = \langle S, A, T, \mathbf{R}, \boldsymbol{\Delta}, o\rangle$ with S denoting the finite set of states, A denoting the finite set of actions, $T: S \times A \times S \rightarrow [0,1]$ is the transition function and $\mathbf{R} = [R_1, ..., R_k]^T$ is the vector of reward functions with $R_i : S \times A \rightarrow \mathbb{R}$, $\boldsymbol{\Delta} = \langle \delta_1, ..., \delta_k\rangle$ is the tuple of *slack* variables with $\delta_i \geq 0$, and o denotes the strict preference ordering over the k objectives. The slack δ_i is an additive value denoting the acceptable deviation from the optimal expected reward for objective o_i so as to improve the lower priority objectives. The set of value functions is denoted by $\boldsymbol{V} = [V_1, ..., V_k]^T$, with V_i denoting the value function corresponding to o_i, and calculated as

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

LMDP sequentially processes each objective in the lexicographic order and therefore the policy of the current objective and the slack determine the actions available for optimizing the next objective. The set of restricted actions for o_{i+1} is:

$$A_{i+1}(s) = \{a \in A | \max_{a' \in A_i} Q_i(s, a') - Q_i(s, a) \leq \eta_i\}$$

where $\eta_i = (1-\gamma)\delta_i$, with a discount factor $\gamma \in [0, 1)$. We refer to [Wray *et al.*, 2015] for a detailed background on LMDP.

3 Problem Formulation

Consider an agent that reasons using its acquired model, an MDP $\tilde{M} = \langle \tilde{S}, \tilde{A}, \tilde{T}, \tilde{R}\rangle$. The agent's model $\tilde{M}$ includes a single objective, which is the primary task of the agent. However, the agent is situated in a more complex environment which is an extension of $\tilde{M}$ and modeled as an LMDP, denoted by M^*, with an additional secondary objective, initially

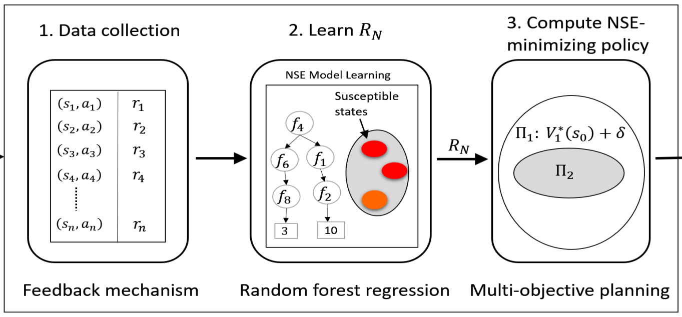


Figure 2: Our framework for mitigating negative side effects.

unknown to the agent. The two objectives in M^* are: assigned task (o_1) and mitigate side effects (o_2), with $o_1 \succ o_2$. We consider a factored state representation for M^* and $\tilde{M}$.

We make the following assumptions about the relationship between $\tilde{M}$ and M^*: (1) $\tilde{M}$ has all the components necessary to achieve o_1 – an optimal policy of $\tilde{M}$ for o_1 is also optimal in M^* with respect to o_1; (2) The agent does not have any prior knowledge about o_2, which reflects the NSE, and its associated penalty denoted by R_N; (3) $\tilde{M}$ may have limited state representation, compared to M^*, regarding o_2; and (4) $\tilde{M}$ and M^* are stationary. The effectiveness of our approach to avoid NSE will depend on the fidelity of $\tilde{S}$. While perfect information is not required, we assume in this work that the occurrence of NSE correlates with the features in $\tilde{S}$. This allows the agent to learn R_N using its current model $\tilde{M}$. For example, the agent can learn to recognize that pushing the box over a rug is undesirable because it is tied to specific locations that are part of its state representation. If the surface type is a state feature, the agent could further generalize what it learns.

Definition 1. *A* ***primary policy*** *is an optimal policy for $\tilde{M}$, optimizing the agent's primary objective defined by $\tilde{R}$.*

Executing a primary policy may result in NSE in some states, since it optimizes for o_1 alone. Let $\Omega : \tilde{S} \times \tilde{A} \to \mathbb{R}$ be a mapping that denotes the severity of the expected NSE produced by executing $\tilde{a}$ in $\tilde{s}$. States in which the agent's actions have immediate NSE ($\Omega(\tilde{s}, \tilde{a}) > 0$) are called *susceptible* states. The agent may not be able to observe the NSE except for the penalty, which is proportional to the severity of the NSE, provided by the feedback mechanism. In the interest of clarity, we assume that executing certain actions in susceptible states always leads to NSE. The formulation naturally applies to situations in which the occurrence of NSE is probabilistic or when a state in $\tilde{M}$ maps to multiple states in the real world with different NSE severities, by calculating an *expected* penalty that accounts for the likelihood.

In this paper, we focus on the problem of identifying and mitigating NSE, without redesigning the entire model, using various feedback mechanisms. There may be limited opportunity for avoiding NSE when optimizing for o_1. Therefore, we consider a slack value δ, which denotes the maximum allowed deviation of a policy from the optimal expected reward for o_1 in order to minimize NSE.

Given $\tilde{M}$ and feedback data regarding side effects, the agent is expected to compute a policy that optimizes o_1, while avoiding NSE, subject to a slack value. Our formulation can hence handle settings with both avoidable and unavoidable negative side effects. To achieve this, the corresponding LMDP of the agent's $\tilde{M}$ is defined by augmenting it with objective o_2 and a penalty function for NSE.

Definition 2. *The* ***augmented MDP*** *of a given model $\tilde{M}$ is a lexicographic MDP, denoted $M = \langle S, A, T, \boldsymbol{R}, o, \delta \rangle$ s.t.:*

- *$S = \tilde{S}$ denotes the state space;*
- *$A = \tilde{A}$ denotes the set of actions;*
- *$T = \tilde{T}$ denotes the transition function;*
- *$\boldsymbol{R} = [\mathrm{R}_1, \mathrm{R}_2]$ with $\mathrm{R}_1 = \tilde{R}$ denotes the reward associated with primary objective of the agent and $\mathrm{R}_2 = R_N$ denotes the reward associated with NSE of the actions;*
- *$o = [o_1, o_2]$ denotes the objectives where o_1 is the primary objective denoting the agent's assigned task and o_2 is minimizing NSE with $o_1 \succ o_2$; and*
- *$\delta \geq 0$ is the maximum slack or deviation from optimal expected reward for o_1 in order to minimize NSE.*

Since we have two objectives and we only impose slack on the primary objective, we use a single slack parameter δ. Since the agent cannot predict the NSE a priori, it must learn R_N using feedback mechanisms (discussed in Section 4).

The framework for minimizing the NSE involves the following three steps (Figure 2):

1. The agent collects data about NSE through various types of oracle feedback or by exploring the environment;
2. A predictive model of NSE is trained using the gathered data to generalize the agent's observations to unseen situations, represented as a reward function R_N;
3. The agent computes a policy π by solving the augmented MDP optimally with the given δ and learned R_N.

3.1 Slack Estimation

The slack denotes the maximum allowed loss in the expected reward of the agent's primary objective in order to minimize NSE. A smaller slack value limits the scope for minimizing NSE. A very high slack can allow the agent to not fulfill o_1 in an attempt to minimize the NSE. Therefore, the slack determines the overall performance of the agent with respect to both its objectives. Typically the slack value is based on user preferences and the general tolerance towards NSE.

Algorithm 1 `Slack Estimation` $(\tilde{M}, \Omega)$

```
1: $\delta \leftarrow \infty$
2: $\tilde{V}_1^*(s_o) \leftarrow$ Solve $\tilde{M}$ optimally with respect to $o_1$
3: Compute NSE-free transition ($\hat{T}$) by disabling all actions
   that result in negative side effects, $\forall(\tilde{s}, \tilde{a}, \tilde{s}')$:
4: $\hat{T}(\tilde{s}, \tilde{a}, \tilde{s}') \leftarrow \begin{cases} \tilde{T}(\tilde{s}, \tilde{a}, \tilde{s}'), & \text{if } \Omega(\tilde{s}, \tilde{a}) = 0 \\ 0, & \text{otherwise} \end{cases}$
5: if solution exists for $\langle \tilde{S}, \tilde{A}, \hat{T}, \tilde{R} \rangle$ with respect to $o_1$ then
6:     $\hat{V}_1^*(s_o) \leftarrow$ Solve $\langle \tilde{S}, \tilde{A}, \hat{T}, \tilde{R} \rangle$ optimally for $o_1$
7:     $\delta \leftarrow |\tilde{V}_1^*(s_o) - \hat{V}_1^*(s_o)|$
8: return $\delta$
```

We present an approach to determine the minimum slack required to avoid NSE altogether, when feasible (once knowledge about the NSE is obtained). Algorithm 1 determines the slack as the difference between the expected reward of the model before and after disabling all the actions that lead to NSE. The expected return for achieving the task from the single start state s_0 and using $\tilde{M}$ is denoted by $\tilde{V}_1^*(s_0)$. The expected reward after disabling all the actions with NSE, denoted by $\hat{V}_1^*(s_0)$, is the maximum reward that can be achieved without causing any NSE, when possible. Thus the difference in values indicates the minimum slack required to avoid NSE, when feasible, and ensures that the slack is in the same unit as the primary objective. A solution may no longer exist after the actions are disabled (Lines 3-4), in which case $\delta = \infty$ is returned, indicating that it is impossible to completely avoid NSE and still accomplish the task.

Proposition 1. *Given Ω, Algorithm 1 calculates the minimum slack required to avoid NSE, while still accomplishing the agent's primary objective, when NSE are avoidable.*

Proof. We prove this by contradiction. Let δ denote the slack returned by Algorithm 1 and δ^* denote the minimum slack required to avoid NSE in a setting where the NSE are avoidable. Let $\delta^* > \delta$. This indicates that Algorithm 1 produced a lower value of slack than required, resulting in NSE during policy execution. This is possible when the model is not solved optimally or if all the actions that lead to NSE have not been disabled. By Lines 2-6 in Algorithm 1, all unacceptable actions are disabled based on Ω and the model is solved optimally. Therefore, $\delta^* \leq \delta$. Let $\delta^* < \delta$. By Lines 2-6 in Algorithm 1, all actions with NSE are disabled and the model is solved optimally. Hence, $\delta^* \not< \delta$. Therefore, $\delta^* = \delta$. □

Slack is specified by the user when NSE are unavoidable or when δ estimated using Algorithm 1 is beyond the user tolerance. If the models are solved approximately, without solution guarantees, Algorithm 1 is not guaranteed to return the minimum slack but our approach still produces a policy that minimizes NSE, given the slack.

4 Learning Negative Side Effects

To learn about NSE, we consider two forms of feedback that correlate with features in $\tilde{S}$: feedback acquired from an oracle and feedback the agent acquires by exploring the environment. The oracle, typically representing human feedback, provides signals about undesirable actions (with respect to NSE) according to M^*. Alternatively, the agent may explore the environment to gather reward signals with respect to NSE.

4.1 Learning from Human Feedback

Random Queries We begin by considering an ideal setting in which the agent randomly selects an (s, a) pair for querying an oracle, given a budget and without replacement, and receives the exact penalty for the state-action pair with respect to NSE. This approach does not introduce any bias as the samples are *i.i.d.*, allowing the agent to learn the true underlying R_N as the budget for querying increases. Hence this approach offers an upper bound for learning a model of NSE.

Despite the benefits offered by this approach, it is often unrealistic to expect exact penalty specification for randomly selected (s, a). Hence we also consider other feedback mechanisms where the oracle (or a human) *approves* the agent's actions, *corrects* the agent's policy, or *demonstrates* an acceptable trajectory. Since such feedbacks do not provide the exact penalty for NSE, feedbacks indicating acceptable actions are mapped to a zero penalty and others are mapped to a fixed, non-zero penalty denoted by k, which in turn contributes to R_N. In our experiments, k is set to the maximum penalty incurred for NSE in the problem.

Approval (HA) The agent randomly selects (s, a) pairs, without replacement, to query the human, who in turn either approves or disapproves the action in that state. The agent learns by mapping the approved actions to zero penalty, $R_N(s, a) = 0$, and the disapproved actions are mapped to a non-zero penalty $R_N(s, a) = k$. We consider two types of human approval: *strict* (HA-S) and *lenient* (HA-L). Strict feedback disapproves all actions that result in NSE. Lenient approval only disapproves actions with severe NSE. The severity threshold for HA-L is a tunable parameter that is problem-specific. Thus with HA-L, the agent will not learn about NSE with low severity and with HA-S, the agent cannot distinguish between actions with different severities of NSE in a state.

Corrections (C) In this form of feedback, the agent performs a trajectory of its primary policy, with the oracle monitoring. If the oracle observes an unacceptable action at any state, it stops the agent and specifies an acceptable action to execute in that state. If all actions in a state lead to NSE, then the oracle specifies an action with the least NSE. The agent proceeds until the goal is reached or until interrupted again. When interrupted, the agent assumes that all actions, except the correction, are unacceptable in that state. If not interrupted, the action is considered to be acceptable. Acceptable actions are mapped to zero penalty, $R_N(s, a) = 0$, and unacceptable actions are mapped to a non-zero penalty, $R_N(s, a) = k$.

Demo-action mismatch (D-AM) In demo-action mismatch, the human provides limited demonstrations. Each demonstration is a trajectory from start to the goal. The agent collects these trajectories and compares them with its primary policy. For all states in which there is an action mismatch, the agent assumes its policy leads to NSE and assigns $R_N(s, a) = k$.

Both C and D-AM introduce bias since the samples are not *i.i.d.* as the visited states are correlated. Furthermore, all actions other than the corrections or those demonstrated are assumed to be unacceptable. Since there may be multiple acceptable actions in each state, this introduces additional bias.

4.2 Learning from Exploration

When feedback from an oracle is expensive to collect or unavailable, it may be easier to allow the agent to identify susceptible states through limited exploration. Learning from exploration uses ϵ-greedy action selection—the agent exploits the action prescribed by its primary policy or explores a random action to learn about NSE. The agent executes an action and observes the corresponding NSE penalty, $R_N(s, a)$. If the exploration is in a simulator designed to learn NSE, then the reward signals only indicate the penalty for NSE. If the agent explores M^*, the reward signals are tuples indicating the reward for the action and NSE penalty.

We consider three exploration strategies: *conservative* – where the agent explores an action with probability 0.1 or follows its primary policy, *moderate* – where the agent either explores an action with probability 0.5 or follows its primary policy, and *radical* – where the agent predominantly explores with probability 0.9, allowing the agent to possibly identify more NSE than the other exploration strategies.

The exploration strategies also suffer from bias induced by correlated samples since the states visited are not *i.i.d.* Note that the agent explores only to learn R_N and the final policy is computed by solving the augmented model.

4.3 Model Learning

The agent's observations are generalized to unseen situations by training a random forest regression (RF) model to predict the penalty R_N. We use the RF model to handle the continuous nature of the penalty and any regression technique may be used in practice. The hyperparameters for the training are determined by a randomized search in the space of RF parameters. For each hyperparameter setting, a three-fold cross validation is performed and the mean squared error is calculated. Parameters with the least mean squared error are selected for training, which is then used to predict the penalty R_N. The augmented MDP is then updated with this learned R_N and the agent computes an NSE-minimizing policy for execution by solving the augmented MDP.

5 Experimental Setup

We perform extensive evaluation of the different feedback mechanisms for mitigating avoidable and unavoidable NSE.

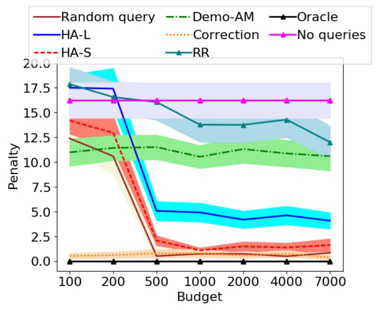
Baselines The performance of our approach is compared with three baselines. First is the *Oracle* agent that avoids NSE while satisfying the primary objective. The *Oracle* has a perfect NSE model and its policy is computed by solving the model after avoiding all the actions with NSE. In problems with unavoidable NSE, it selects the action with the least NSE since that is the best possible performance that can be achieved while satisfying the primary objective. The performance of the *Oracle* provides a lower bound on the penalty for NSE incurred by the agent. The second is the *No queries* case in which the agent does not query or learn about NSE. Instead, it naively executes its primary policy. This provides an upper bound on the penalty for NSE incurred by the agent. Third is the scalarization approach [Krakovna *et al.*, 2019] (*RR*) in which the agent optimizes $r(s_t, a_t)–\beta d(s_t, b_t)$, where $r(s_t, a_t)$ is the reward corresponding to o_1 and $d(s_t, b_t)$ is the measure of deviation from the baseline state b_t, denoting the NSE. A direct comparison with this approach is not feasible since it is based on assumptions that do not hold in our setting. Therefore, we modified the *RR* approach to make it applicable in our setting—by calculating the deviation based on a model of NSE learned with Random Query approach, as it does not introduce any bias. We compute the deviation from inaction baseline, which measures the NSE of the agent's action with respect to no action execution in that state [Krakovna *et al.*, 2019]. The side effects we consider are irreversible by an agent, once occurred, making the baseline state unreachable. We tested with $\beta \in [0.1, 0.9]$ since o_1 is prioritized in our formulation and report results with $\beta = 0.8$ as it achieved the best trade-off in training.

Side Effects In the interest of clarity, we consider two types of NSE severity: mild and severe. Each action can either result in a mild NSE, severe NSE, or no NSE. The strict human approval (HA-S) feedback disapproves all actions that result in NSE. The lenient human approval (HA-L) only disapproves actions with severe NSE. For learning NSE by exploration, we consider agent exploration in a simulator where the reward signals indicate NSE penalty.

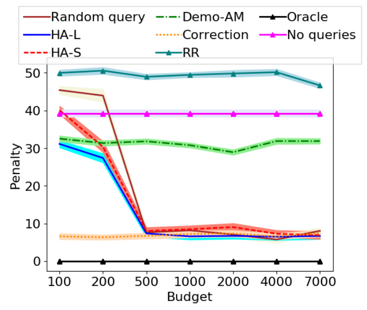
In our experiments, we optimize costs, which are negations of the rewards. Random forest regression from `sklearn` Python package is used for model learning. The augmented MDP is solved using a lexicographic variant of LAO* [Hansen and Zilberstein, 2001]. The slack is computed using Algorithm 1 and $\gamma = 0.95$. Values averaged over 100 trials of planning and execution, along with their standard errors, are reported for the following domains.

Boxpushing We consider a modified boxpushing domain [Seuken and Zilberstein, 2007] in which the agent is expected to minimize the expected time taken to push a box to the goal. Each action takes $+1$ unit of time. Each state is represented as $\langle x, y, b, c\rangle$ where x, y denote the agent's location, b indicates if the agent is pushing the box, c indicates the current cell's surface type. Pushing the box on a surface type $c = 1$ results in severe NSE with a penalty of 10, pushing the box on a surface $c = 2$ results in mild NSE and a penalty of 5, and no NSE otherwise. The state features used for training are $\langle b, c\rangle$. We test on five instances with grid size 15×15 and with varying initial location of the box and the NSE regions.

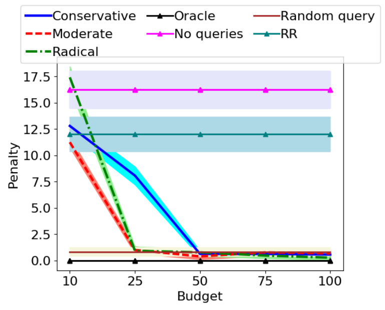
Driving Our second domain is based on simulated autonomous driving [Saisubramanian *et al.*, 2020; Wray *et al.*, 2015] in which the agent's primary objective is to minimize the expected cost of navigation from start to a goal, during which it may encounter some puddles. The agent can move in all four directions at low and high speeds. The cost of navigating at a low speed is 2 and that of high speed is 1. When the agent navigates over a puddle at high speed, it spatters water which is undesirable. Some puddles may have pedestrians in the vicinity and splashing water on them results in severe



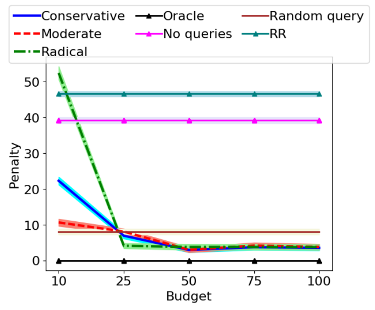
(a) Boxpushing



(b) Driving



(c) Boxpushing



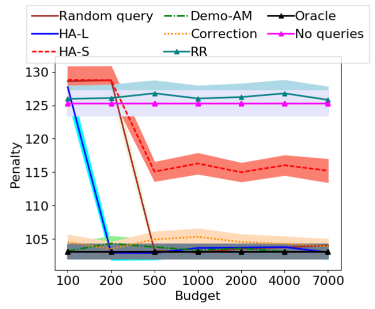
(d) Driving

Figure 3: Effect of learning from human feedback methods (a-b) and with exploration strategies (c-d) on problems with avoidable NSE.

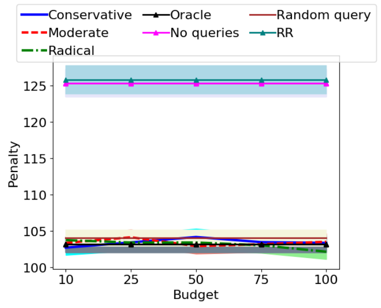
NSE with a penalty of 10 and a mild NSE otherwise with a penalty of 5. Each state is represented by $\langle x, y, l, p, h\rangle$ where x, y denote the agent's location, l is the speed, p, h indicate the presence of a puddle and a pedestrian in the vicinity. Features used for training are $\langle l, p, h\rangle$. Five test instances are generated with grid size 15×15 and by varying the start, goal, puddle, and pedestrian locations.

6 Results and Discussion

Effectiveness of Feedback Mechanisms Figure 3 show the average penalty incurred for NSE in settings with avoidable NSE, both mild and severe, as the training budget is increased. While the Corrections feedback is efficient in terms of samples required, it is more expensive to collect this feedback since it requires constant oversight. Random querying and HA-S, which rely on random samples of states, achieve significant reduction in NSE with 500 samples. Although HA-L also relies on random sampling of states, it does not provide information about mild NSE, which affects its performance. Training with Demo-AM feedback does not always minimize NSE even as the budget is increased, since the agent does not receive enough negative samples and has no information about the NSE in states unvisited in the demonstrations. Additionally, it is unable to distinguish between the different levels of severity of the NSE of its actions. However, Demo-AM is still better than *No queries*. *RR* with the model of NSE learned using Random Query and with the given budget, performs poorly irrespective of the budget. Apart from the reported results, we also tested *RR* with a perfect model of NSE and its performance was significantly better and sometimes comparable to the *Oracle*'s performance. However, obtaining a perfect model of NSE is non-trivial in practice. In Figure 3(c-d), Random querying with the maximum budget (7000) and *RR* with this learned model are plotted in to compare the performance of exploration strategies and to understand how the correlated samples affect the performance. Ir-



(a) Human feedback



(b) Exploration

Figure 4: Performance on problems with unavoidable NSE.

respective of the exploration type, the agent learns the NSE with a reasonable number of trials. Figure 6(a) shows the performance of the approaches with respect to o_1.

Unavoidable NSE Since Algorithm 1 does not produce a finite slack for this setting, we experiment with a slack value that is 15% of the expected cost of o_1. This value is based on the observation that slack values returned by Algorithm 1 are within 15% of the expected cost for o_1, for most problem instances with avoidable NSE. Figure 4 plots the results for boxpushing problems with unavoidable NSE. Problem instances with unavoidable NSE case were generated by making sure there is no path to the goal over surface $c = 3$. Since HA-S cannot distinguish between different severities, its performance is affected when NSE are unavoidable. Unlike Figure 3(a-b), Demo-AM matches the performance of other techniques when NSE are unavoidable. Similar to Figure 3(c-d), learning by exploration in settings with unavoidable NSE matches the performance of Random query. Similar results were obtained for driving domain. Overall, the results indicate that our framework can effectively learn and mitigate the impacts of both avoidable and unavoidable NSE.

Bias of Various Feedback Mechanisms Figure 5 plots the frequency of querying in different regions of the state space with different feedback mechanisms. The x-axis denotes the state space and darker shades indicate regions that were frequently queried. The results are plotted for driving domain with avoidable NSE and using 7000 queries for human feedback and 100 trials for exploration. Feedback techniques based on random sampling of states have a higher coverage of the state space, contributing to a better performance. The exploration techniques are the most restricted, due to which their performances are largely similar. Since an ϵ-greedy approach is followed for exploration, these techniques likely cover only the region surrounding that of the primary policy. As states in this region are often critical for satisfying the agent's primary objective, learning about NSE in this restricted region is often sufficient to effectively mitigate the penalty for NSE. This result shows that different feedback types focus on different regions of the state space, which in turn affects the NSE model learning.

Effect of Slack We study the effect of slack on the expected cost of o_1 and on NSE (o_2), by varying the slack from 40% to 100% of the value returned by Algorithm 1 on problems with avoidable NSE. Figure 6 shows the results on the boxpushing domain with 7000 queries for human feedback and 100 exploration trials. The values show the average cost in 100 trials of executing the updated policy with each slack value. We do not compare with the baselines, which are unaffected by the variation in slack. Results with no slack bound the performance of other techniques. As the slack is increased, the average penalty incurred for NSE tends to decrease. The minimal differences in the average costs for o_1 shows that the slack values returned by Algorithm 1 are reasonable and affect o_1 only by a small margin. The performance of some feedback approaches are unaffected by the variation in slack, which shows their limitations in minimizing NSE.

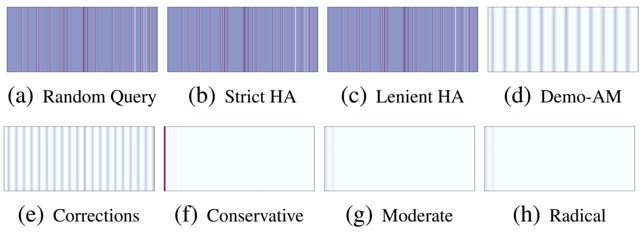
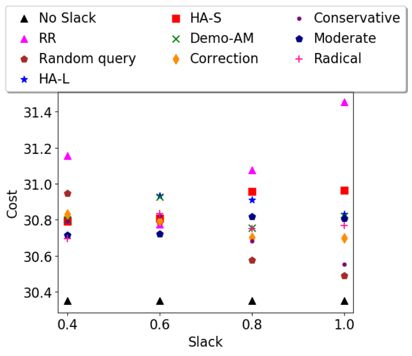
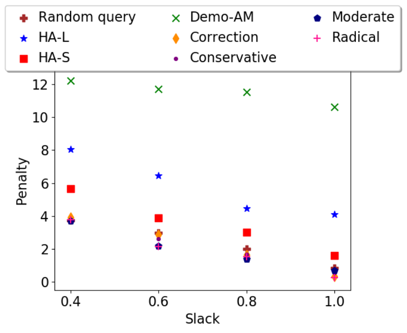

(a) Random Query (b) Strict HA (c) Lenient HA (d) Demo-AM (e) Corrections (f) Conservative (g) Moderate (h) Radical

Figure 5: Bias in feedback techniques.



(a) Effect on o_1



(b) Effect on o_2

Figure 6: Effect of slack on the performance.

7 Conclusion and Future Work

We formulate the problem of mitigating the immediate NSE of actions in a state as a multi-objective problem with slack and propose an algorithm to determine the minimum slack required to avoid these side effects. Various feedback mechanisms are considered for learning a model of negative side effects and their biases are analyzed empirically. Our proposed framework is shown to be effective in mitigating undesirable side effects. A key advantage of our approach is that it allows learning about NSE during agent deployment, without the need to suspend operation to allow redesign of the reward function. In the future, we aim to extend our approach to settings in which the side effects are partially observable and to settings where effectively avoiding the side effects requires adding new features to the state representation.

Acknowledgments

This work was supported by the Semiconductor Research Corporation grant #2906.001.

References

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

[Dietterich, 2017] Thomas G. Dietterich. Steps toward robust artificial intelligence. *AI Magazine*, 38(3):3–24, 2017.

[Hadfield-Menell *et al.*, 2017] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J. Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems*, 2017.

[Hansen and Zilberstein, 2001] Eric A. Hansen and Shlomo Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129:35–62, 2001.

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

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

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

[Saisubramanian *et al.*, 2019] Sandhya Saisubramanian, Kyle Wray, Luis Pineda, and Shlomo Zilberstein. Planning in stochastic environments with goal uncertainty. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[Saisubramanian *et al.*, 2020] Sandhya Saisubramanian, Ece Kamar, and Shlomo Zilberstein. Mitigating the negative side effects of reasoning with imperfect models: A multi-objective approach. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2020.

[Seuken and Zilberstein, 2007] Sven Seuken and Shlomo Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

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

[Zhang *et al.*, 2018] Shun Zhang, Edmund H. Durfee, and Satinder P. Singh. Minimax-regret querying on side effects for safe optimality in factored Markov decision processes. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[Zilberstein, 2015] Shlomo Zilberstein. Building strong semi-autonomous systems. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI)*, 2015.