Mitigating Negative Side Effects via Environment Shaping

Extended Abstract

Sandhya Saisubramanian
University of Massachusetts Amherst

Shlomo Zilberstein
University of Massachusetts Amherst

ABSTRACT

Agents operating in the open world often produce *negative side effects* (NSE), which are difficult to identify at design time. We examine how a human can assist an agent, beyond providing feedback, and exploit their broader scope of knowledge to mitigate the impacts of NSE. We formulate this problem as a human-agent team with decoupled objectives. The agent optimizes its assigned task, during which its actions may produce NSE. The human shapes the environment through minor reconfiguration actions so as to mitigate the impacts of agent's side effects, without significantly degrading agent performance. We present an algorithm to solve this problem. Empirical evaluation shows that the proposed framework can successfully mitigate NSE, without affecting the agent's ability to complete its assigned task.

KEYWORDS

Negative Side Effects; Environment Shaping; Human-AI Teamwork

ACM Reference Format:
Sandhya Saisubramanian and Shlomo Zilberstein. 2021. Mitigating Negative Side Effects via Environment Shaping: Extended Abstract. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 3 pages.

1 INTRODUCTION

Deploying AI systems requires complex design choices to support safe operation in the open world. During the design and initial testing, the system designer typically ensures that the agent's model includes all the necessary details relevant to its assigned task. Inherently, many other details of the environment that are unrelated to this task may be ignored. Due to this model incompleteness, the agent's actions may create *negative side effects* (NSE) [1, 6]. Mitigating such NSE is critical to improve trust in deployed AI systems. However, it is practically impossible to identify all such NSE at design time since agents are deployed in varied settings. Deployed agents often do not have any prior knowledge about NSE, and therefore they do not have the ability to minimize NSE. *How can we leverage human assistance and their broader scope of knowledge to mitigate the negative side effects, when agents are unaware of the side effects and the associated penalties?*

A common solution approach in the existing literature is to update the agent's model by learning about NSE through feedback [2, 5, 7]. This approach has three main drawbacks: (1) the NSE may be non-Markovian with respect to the agent's limited state representation—containing only the features related to its task; (2) extensive model revisions are costly and will likely require exhaustive evaluation before the system can be redeployed; and (3) does not mitigate the impacts when the NSE are unavoidable.

The key insight of this paper is that agents often operate in environments that are configurable, which can be leveraged to mitigate NSE. We propose ***environment shaping*** for deployed agents: a process of applying simple modifications to the current environment to make it more agent-friendly and minimize the occurrence of NSE. Simple modifications to the environment have shown to accelerate agent learning [4] and goal recognition [3]. We target settings in which environment shaping can reduce NSE, without significantly degrading the performance of the agent in completing its assigned task. Our formulation consists of an *actor* and a *designer* with decoupled objectives. The actor agent computes a policy that optimizes the completion of its assigned task and has no knowledge about NSE of its actions. The designer agent shapes the environment through minor modifications so as to mitigate the NSE of actor's actions, without affecting the actor's ability to complete its assigned task.

2 ACTOR-DESIGNER FRAMEWORK

Consider an *actor* agent that operates based on a Markov decision process (MDP) M_a in an environment that is configurable and described by a configuration file E, such as a map. A factored state representation is assumed. Executing the policy π, computed using M_a, to complete its assigned task o_P may lead to NSE, unknown to the actor. The *environment designer* measures the impact of NSE associated with the actor's π and shapes the environment, if necessary. Optimizing o_P is prioritized over avoiding NSE. Hence, the designer shapes the environment *in response* to π. Each modification is a sequence of design actions. An example is $\{move(table, l_1, l_2), remove(rug)\}$, which moves the table from l_1 to l_2 and removes the rug, resulting in a new environment configuration. The actor and the environment designer share the configuration file of the environment, which is updated by the designer to reflect the modifications. We make the following assumptions about shaping for NSE: (1) NSE are undesirable but not safety-critical, and its occurrence does not affect the actor's ability to complete its task; (2) the start and goal conditions of the actor are fixed and cannot be altered; and (3) modifications are applied tentatively for evaluation purposes and the environment is reset if the reconfiguration affects the actor's ability to complete its task or if the modification does not minimize the NSE.

Definition 1. *An actor-designer framework to mitigate negative side effects* (**AD-NSE**) *is defined by* $\langle E_0, \mathcal{E}, M_a, M_d, \delta_A, \delta_D \rangle$ *with:*

- E_0 *denoting the initial environment configuration of the actor;*
- $\mathcal{E}$ *denoting a finite set of possible reconfigurations of* E_0*;*
- $M_a = \langle S, A, T, R, s_0, s_G \rangle$ *is the actor's MDP with start state* s_0 *and goal state* s_G*, corresponding to its task* o_P*;*

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

- *$M_d = \langle \Omega, \Psi, C, N \rangle$ is the model of the designer with Ω denoting a finite set of valid modifications for E_0, including $\emptyset$ to indicate that no changes are made; $\Psi : \mathcal{E} \times \Omega \rightarrow \mathcal{E}$ determines the resulting environment configuration after applying a modification $\omega \in \Omega$ and is denoted by $\Psi(E, \omega)$; $C : \mathcal{E} \times \Omega \rightarrow \mathbb{R}$ specifies the cost of applying a modification to an environment, with $C(E, \emptyset) = 0, \forall E \in \mathcal{E}$; and $N = \langle \pi, E, \zeta \rangle$ is a model specifying the penalty for NSE in environment $E \in \mathcal{E}$ for the actor's policy π, with ζ mapping states in π to E for severity estimation;*
- *$\delta_A \geq 0$ is the actor's slack, denoting the maximum allowed deviation from the initial optimal policy when recomputing its policy in a modified environment; and*
- *$\delta_D \geq 0$ indicates the designer's tolerance threshold for NSE.*

The actor and the designer do not have knowledge about each other's model. The designer observes a finite number of demonstrations of the actor's policy, $\mathcal{D} = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Using $\mathcal{D}$, the designer extracts the actor's policy and measure its NSE in the current environment configuration E, denoted by N_π^E. The environment is modified if $N_\pi^E > \delta_D$, assuming π is fixed. The designer selects a modification that maximizes its utility, calculated as:

$$U_\pi(\omega) = \left(N_\pi^E - N_\pi^{\Psi(E,\omega)}\right) - C(E, \omega),$$

with $N_\pi^E - N_\pi^{\Psi(E,\omega)}$ denoting the reduction in NSE. The designer updates the configuration file to reflect the modifications, which is used by the actor to recompute its policy. Apart from $\mathcal{D}$ and the environment configuration, no other knowledge is shared between the actor and the designer. It is assumed that the cost of applying an ω is in the same units as the NSE penalty, and it may be amortized over episodes of the actor performing the task in the environment.

3 SOLUTION APPROACH

Our solution approach is outlined in Figure 1 and proceeds in two phases: planning phase and shaping phase. In the planning phase, the actor computes a policy π for o_P in the current environment configuration and provides limited demonstrations $\mathcal{D}$, which are trajectories from start to goal. In the shaping phase (indicated by blue symbols in Figure 1), the designer first extracts a policy $\hat{\pi}$ from $\mathcal{D}$ by associating states with actions observed. Then, the corresponding NSE penalty, $N_{\hat{\pi}}^E$, is estimated. If $N_{\hat{\pi}}^E > \delta_D$, the designer applies a utility-maximizing modification and updates the configuration file, which is then used by the actor to recompute its policy. The modifications are *tentative* during the shaping and evaluation, and reset if it affects the actor's performance or does not minimize NSE. Therefore, all modifications are applied to E_0 and it suffices to test each ω without replacement as the actor always

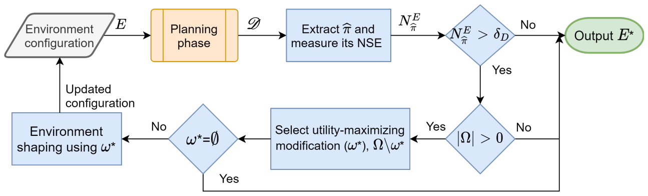
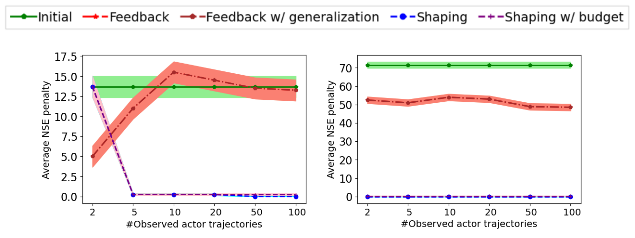

Environment configuration
E
Planning phase
$\mathcal{D}$
Extract $\hat{\pi}$ and measure its NSE
$N_{\hat{\pi}}^E$
$N_{\hat{\pi}}^E > \delta_D$
No
Output E^*
Yes
Updated configuration
Environment shaping using ω^*
No
$\omega^* = \emptyset$
Select utility-maximizing modification (ω^*), $\Omega \setminus \omega^*$
Yes
$|\Omega| > 0$
No
Yes

Figure 1: Overview of our solution approach to mitigate NSE via environment shaping.



(a) Avoidable NSE (b) Unavoidable NSE

Figure 2: Results on boxpushing domain with avoidable and unavoidable NSE.

calculates the corresponding optimal policy. The actor returns $\mathcal{D} = \{\}$ when the modification affects its assigned task, given δ_A.

The planning and shaping phases alternate until one of the following conditions is met: (1) the NSE is within δ_D; (2) all $\omega \in \Omega$ have been tested; or (3) the utility-maximizing option does not make any modification at all, $\omega^* = \emptyset$. Upon termination, the algorithm outputs an NSE-minimizing configuration of the environment.

In shaping with budget, the budget $0 < b \leq |\Omega|$ denotes the maximum number of modifications the designer is willing to evaluate. In this case, b diverse modifications are selected for evaluation in the shaping phase, instead of using the entire Ω. Diverse modifications are selected by comparing all pairs of ω. When two modifications result in similar configurations, we prune the modification with a higher cost. The similarity threshold for pruning is a tunable parameter. Measures such as the Jaccard distance or embeddings may be used to estimate the similarity between two configurations.

4 RESULTS AND FUTURE WORK

The performance of our approach is compared with: (1) *Initial* approach in which the actor's policy is naively executed and does not involve shaping or any form of learning to mitigate NSE; and (2) *Feedback* approach in which the human approves or disapproves agent actions [5]. The budget for feedback is 500. In *Feedback w/ generalization*, the actor generalizes the human feedback to unseen situations by learning a predictive model of NSE occurrence. We report results on the boxpushing domain in which the actor is required to minimize the expected time taken to push a box to the goal location. 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. Pushing the box over the rug results in NSE with a penalty of 5. We experiment with a grid of size 15×15. We consider 24 modifications, such as adding a protective sheet over the rug, removing the rug, block access to the rug, among others. Figure 2 plots the results with $\delta_A = 25\%$ of the optimal expected cost of o_P, $\delta_D = 0$, and $b = 3$, as the number of observed actor trajectories is increased. Our results demonstrate the effectiveness of shaping in mitigating NSE. In the future, we aim to develop techniques to automatically identify and efficiently explore a large space of valid modifications.

ACKNOWLEDGMENTS

Support for this work was provided in part by the Semiconductor Research Corporation under grant #2906.001.

REFERENCES

[1] 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).

[2] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J. Russell, and Anca Dragan. 2017. Inverse Reward Design. In *Advances in Neural Information Processing Systems*. 6765–6774.

[3] Sarah Keren, Avigdor Gal, and Erez Karpas. 2014. Goal Recognition Design. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling*.

[4] Jette Randløv. 2000. Shaping in Reinforcement Learning by Changing the Physics of the Problem. In *Proceedings of the 17th International Conference on Machine Learning*. 767–774.

[5] Sandhya Saisubramanian, Ece Kamar, and Shlomo Zilberstein. 2020. A Multi-Objective Approach to Mitigate Negative Side Effects. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. 354–361.

[6] Sandhya Saisubramanian, Shlomo Zilberstein, and Ece Kamar. 2020. Avoiding Negative Side Effects due to Incomplete Knowledge of AI Systems. *CoRR* abs/2008.12146 (2020).

[7] Shun Zhang, Edmund H. Durfee, and Satinder P. Singh. 2018. 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*. 4867–4873.