Penalizing side effects using stepwise relative reachability

Victoria Krakovna*, **Laurent Orseau**, **Miljan Martic** and **Shane Legg**
DeepMind

Abstract

How can we design safe reinforcement learning agents that avoid unnecessary disruptions to their environment? We show that current approaches for penalizing side effects are ineffective in the case where achieving the objective requires irreversible actions. They can also introduce bad incentives for the agent: interference, where the agent prevents any irreversible changes in the environment (including the actions of other agents), and offsetting, where the agent undoes its own actions towards the objective. To isolate the source of these failure modes, we break down side effects penalties into two independent components: a baseline state and a measure of deviation from this baseline state. We argue that the interference and offsetting incentives arise from the choice of baseline state, while the choice of deviation measure determines the effectiveness of the penalty at avoiding side effects. We introduce a new deviation measure based on relative reachability of states that penalizes side effects when the simpler unreachability measure fails. We also show that the stepwise inaction baseline (where the agent does nothing instead of its last action) avoids the bad incentives where other baselines fail. We empirically compare different combinations of baseline and deviation measure choices on a set of gridworld experiments designed to illustrate possible failure modes, and show that only the combination of the relative reachability measure with the stepwise inaction baseline avoids all the failure modes simultaneously.

1 Introduction

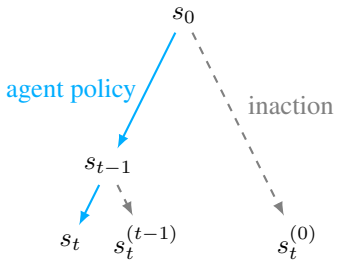
An important component of safe behavior for reinforcement learning agents is avoiding unnecessary side effects while performing a task [Amodei *et al.*, 2016; Taylor *et al.*, 2016]. For example, if an agent's task is to carry a box across the room, we want it to do so without breaking vases, while an agent tasked with eliminating a computer virus should avoid unnecessarily deleting files. The side effects problem is related to the frame problem in classical AI [McCarthy and Hayes, 1969]. For machine learning systems, it has mostly been studied in the context of safe exploration during the agent's learning process [Pecka and Svoboda, 2014; García and Fernández, 2015], but can also occur after training if the reward function is misspecified and fails to penalize disruptions to the environment [Ortega *et al.*, 2018].

We would like to incentivize the agent to avoid side effects without explicitly penalizing every possible disruption, defining disruptions in terms of predefined state features, or going through a process of trial and error when designing the reward function. While such approaches can be sufficient for agents deployed in a narrow set of environments, they often require a lot of human input and are unlikely to scale well to increasingly complex and diverse environments. It is thus important to develop more general and systematic approaches for avoiding side effects.

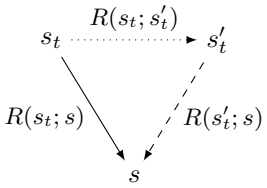
Most of the general approaches to this problem are reachability-based methods: methods that preserve reversibility (the reachability of a starting state) [Moldovan and Abbeel, 2012; Eysenbach *et al.*, 2017], and reachability analysis methods that require reachability of a safe region [Mitchell *et al.*, 2005; Gillula and Tomlin, 2012; Fisac *et al.*, 2017]. The reachability criterion has a notable limitation: it is insensitive to the magnitude of the irreversible disruption, e.g. it equally penalizes the agent for breaking one vase or a hundred vases. Thus, this criterion is ineffective for avoiding side effects if the objective requires an irreversible action. Comparison to a starting state also introduces undesirable incentives in *dynamic* environments, where irreversible transitions can happen spontaneously (due to the forces of nature, the actions of other agents, etc). Since such transitions make the starting state unreachable, the agent has an incentive to interfere to prevent them. This is often undesirable, e.g. if the transition involves a human eating food. Thus, while these methods address the side effects problem in environments where the agent is the only source of change and the objective does not require irreversible actions, a more general criterion is needed when these assumptions do not hold.

The contributions of this paper are as follows. In Section 2, we introduce a breakdown of side effects penalties into two design choices, a baseline state and a measure of deviation of the current state from the baseline state, as shown in Figure 1. In Section 2.1, we introduce two types of bad incentives (inter-

*Contact Author, vkrakovna@google.com



(a) Choices of baseline state s'_t: starting state s_0, inaction $s_t^{(0)}$, and stepwise inaction $s_t^{(t-1)}$. Actions drawn from the agent policy are shown by solid blue arrows, while actions drawn from the inaction policy are shown by dashed gray arrows.



(b) Choices of deviation measure d: given a function $R(x; y)$ that defines the reachability of y from x, $d_{UR}(s_t; s'_t) := 1 - R(s_t; s'_t)$ is the unreachability measure of the baseline state s'_t from the current state s_t (dotted line), while the relative reachability measure $d_{RR}(s_t; s'_t) := \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \max(R(s'_t; s) - R(s_t; s), 0)$ is the average reduction in reachability of states s from current state s_t (solid line) compared to the baseline state s'_t (dashed line).

Figure 1: Design choices for a side effects penalty: baseline states and deviation measures.

ference and offsetting), along with environments that test for them. We argue that they arise from the choice of baseline and show that the stepwise inaction baseline (shown in Figure 1a) avoids them. In Section 2.2, we show that the unreachability measure is insensitive to the magnitude of the agent's effects, and propose a magnitude-sensitive *relative reachability* measure, defined by comparing the reachability of states between the current state and the baseline state, as shown in Figure 1b. In Section 3, we compare all combinations of the baseline and deviation measure choices on gridworld environments that test for side effects and bad incentives, and show that relative reachability with the stepwise inaction baseline is the only combination that passes on all the environments.

We do not claim this approach to be a complete solution to the side effects problem, since there may be other cases of bad behaviors that we have not considered. However, we believe that avoiding the bad behaviors we described is a bare minimum for an agent to be both safe and useful, so our approach provides some necessary ingredients for a solution to the problem.

Preliminaries. We assume that the environment is a discounted Markov Decision Process (MDP), defined by a tuple $(\mathcal{S}, \mathcal{A}, r, p, \gamma)$. $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $p(s_{t+1}|s_t, a_t)$ is the transition function, and $\gamma \in (0, 1)$ is the discount factor. At time step t, the agent receives the state s_t, outputs the action a_t drawn from its policy $\pi(a_t|s_t)$, and receives reward $r(s_t, a_t)$. We define a transition as a tuple (s_t, a_t, s_{t+1}) consisting of state s_t, action a_t, and next state s_{t+1}. We assume that there is a special *noop* action a^{noop} that has the same effect as the agent being turned off during the given time step. We denote $s_t^{(k)}$ as the state obtained by starting in state s_k and taking the noop action $t - k$ times until time step t.

Intended effects and side effects. We begin with some motivating examples for distinguishing intended and unintended disruptions to the environment:

Example 1 (Vase)**.** The agent's objective is to get from point A to point B as quickly as possible, and there is a vase in the shortest path that would break if the agent walks into it.

Example 2 (Omelette)**.** The agent's objective is to make an omelette, which requires breaking eggs.

In both of these cases, the agent would take an irreversible action by default (breaking a vase vs breaking eggs). However, the agent can still get to point B without breaking the vase (at the cost of a bit of extra time), but it cannot make an omelette without breaking eggs. We would like to incentivize the agent to avoid breaking the vase while allowing it to break the eggs.

Safety criteria are often implemented as constraints [García and Fernández, 2015; Moldovan and Abbeel, 2012; Eysenbach *et al.*, 2017]. This approach works well if we know exactly what the agent must avoid, but is too inflexible for a general criterion for avoiding side effects. For example, a constraint that the agent must never make the starting state unreachable would prevent it from making the omelette in Example 2, no matter how high the reward for doing so.

A more flexible way to implement a side effects criterion is by adding a penalty for impacting the environment to the reward function, which acts as an intrinsic pseudo-reward. An impact penalty at time t can be defined as a measure of *deviation* of the current state s_t from a *baseline* state s'_t, denoted as $d(s_t; s'_t)$. Then at every time step t, the agent receives the following total reward:

$$r(s_t, a_t) - \beta \cdot d(s_{t+1}; s'_{t+1}).$$

Since the task reward r indicates whether the agent has achieved the objective, we can distinguish intended and unintended effects by balancing the task reward and the penalty using the scaling parameter β. Here, the penalty would outweigh the small reward gain from walking into the vase over going around the vase, but it would not outweigh the large reward gain from breaking the eggs.

2 Design choices for an impact penalty

When defining the impact penalty, the baseline s'_t and deviation measure d can be chosen separately. We will discuss several possible choices for each of these components.

2.1 Baseline states

Starting state baseline. One natural choice of baseline state is the starting state $s'_t = s_0$ when the agent was deployed (or a starting state distribution), which we call the *starting state baseline*. This is the baseline used in reversibility-preserving approaches, where the agent learns a reset policy that is rewarded for reaching states that are likely under the initial state distribution.

While penalties with the starting state baseline work well in environments where the agent is the only source of change, in dynamic environments they also penalize irreversible transitions that are not caused by the agent. This incentivizes the agent to interfere with other agents and environment processes to prevent these irreversible transitions. To illustrate this *interference* behavior, we introduce the Conveyor Belt Sushi environment, shown in Figure 2.

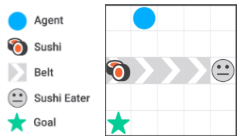


Figure 2: Sushi environment.

This environment is a sushi restaurant, which contains a conveyor belt that moves to the right by one square after every agent action. There is a sushi dish on the belt that is eaten by a human if it reaches the end of the belt. The interference behavior is to move the sushi off the belt. The agent is rewarded for reaching the goal square, and it can reach the goal with or without interfering with the sushi in the same number of steps. The desired behavior is to reach the goal without interference, by going left and then down. As shown in Section 3, impact penalties with the starting state baseline produce the interference behavior.

Inaction baseline. Another choice is the *inaction* baseline $s'_t = s_t^{(0)}$: a counterfactual state of the environment if the agent had done nothing for the duration of the episode. Inaction can be defined in several ways. Armstrong and Levinstein [2017] define it as the agent never being deployed: conditioning on the event X where the AI system is never turned on. It can also be defined as following some baseline policy, e.g. a policy that always takes the noop action a^{noop}. We use this noop policy as the inaction baseline in this work. Penalties with this baseline do not produce the interference behavior in dynamic environments, since transitions that are not caused by the agent would also occur in the inaction counterfactual and thus are not penalized.

However, the inaction baseline incentivizes another type of undesirable behavior, called *offsetting*. We introduce the Conveyor Belt Vase environment to illustrate this behavior, shown in Figure 3.

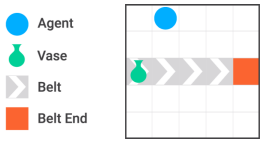
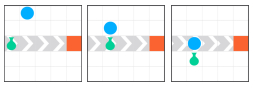
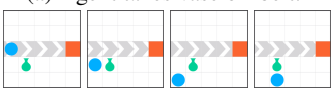


Figure 3: Vase environment.

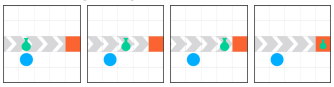
This environment also contains a conveyor belt, with a vase that will break if it reaches the end of the belt. The agent receives a reward for taking the vase off the belt. The desired behavior is to move the vase off and then stay put. The offsetting behavior is to move the vase off (thus collecting the reward) and then put it back on, as shown in Figure 4.



(a) Agent takes vase off belt.



(b) Agent goes around vase.



(c) Agent puts vase back on belt.

Figure 4: Offsetting behavior in the Vase environment.

Offsetting happens because the vase breaks in the inaction counterfactual. Once the agent takes the vase off the belt, it continues to receive penalties for the deviation between the current state and the baseline. Thus, it has an incentive to return to the baseline by breaking the vase after collecting the reward. Experiments in Section 3 show that impact penalties with the inaction baseline produce the offsetting behavior if they have a nonzero penalty for taking the vase off the belt.

Stepwise inaction baseline. The inaction baseline can be modified to branch off from the previous state s_{t-1} rather than the starting state s_0. This is the *stepwise inaction* baseline $s'_t = s_t^{(t-1)}$: a counterfactual state of the environment if the agent had done nothing instead of its last action [Turner *et al.*, 2019]. This baseline state is generated by a baseline policy that follows the agent policy for the first $t-1$ steps, and takes an action drawn from the inaction policy on step t. Each transition is penalized only once, at the same time as it is rewarded, so there is no offsetting incentive.

However, there is a problem with directly comparing current state s_t with $s_t^{(t-1)}$: this does not capture delayed effects of action a_{t-1}. For example, if this action is putting a vase on a conveyor belt, then the current state s_t contains the intact vase, and by the time the vase breaks, the broken vase will be part of the baseline state. Thus, the penalty for action a_{t-1} needs to be modified to take into account future effects of this action, e.g. by using *inaction rollouts* from the current state and the baseline (Figure 5).

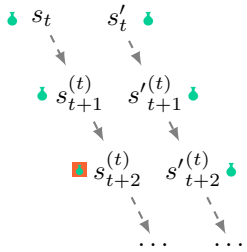


Figure 5: Inaction rollouts from the current state s_t and baseline state s'_t used for penalizing delayed effects of the agent's actions. If action a_{t-1} puts a vase on a conveyor belt, then the vase breaks in the inaction rollout from s_t but not in the inaction rollout from s'_t.

An inaction rollout from state $\tilde{s}_t \in \{s_t, s'_t\}$ is a sequence of states obtained by following the inaction policy starting from that state: $\tilde{s}_t, \tilde{s}^{(t)}_{t+1}, \tilde{s}^{(t)}_{t+2}, \ldots$. Future effects of action a_{t-1} can be modeled by comparing an inaction rollout from s_t to an inaction rollout from $s^{(t-1)}_t$. For example, if action a_{t-1} puts the vase on the belt, and the vase breaks 2 steps later, then $s^{(t)}_{t+2}$ will contain a broken vase, while $s'^{(t)}_{t+2}$ will not. Turner *et al.* [2019] compare the inaction rollouts $s^{(t)}_{t+k}$ and $s'^{(t)}_{t+k}$ at a single time step $t+k$, which is simple to compute, but does not account for delayed effects that occur after that time step. We will introduce a recursive formula for comparing the inaction rollouts $s^{(t)}_{t+k}$ and $s'^{(t)}_{t+k}$ for all $k \geq 0$ in Section 2.2.

2.2 Deviation measures

We will consider reachability-based deviation measures. We define reachability of state y from state x as the value function of the optimal policy given a reward of 1 for reaching y and 0 otherwise:

$$R(x; y) := \max_{\pi} \mathbb{E}\, \gamma_r^{N_\pi(x;y)}$$

where $N_\pi(x; y)$ is the number of steps it takes to reach y from x when following policy π, and $\gamma_r \in (0, 1]$ is the reachability discount factor. A special case is *undiscounted* reachability ($\gamma_r = 1$), which computes whether y is reachable in any number of steps.

Without a full environment model, reachability can be computed dynamically as the agent explores the environment, based on states and transitions that the agent has encountered (assuming a deterministic environment). Reachability is initialized as $R(x; y) = 1$ if $x = y$ and 0 otherwise (different states are unreachable from each other). When the agent makes a new transition (s_t, a_t, s_{t+1}), we update the reachability function to consider all shortest paths that involve this transition: for all pairs of states x and y, we set $R(x; y) = \max(R(x; y), R(x; s_t)\gamma_r R(s_{t+1}; y))$, which has time complexity $O(S^2)$ where S is the number of states. If the agent has taken all the edges along the shortest path from x to y, then the computed value is exact, so the greedy approximation converges to true reachability function once each edge has been explored once. The total time complexity is $O(ES^2)$ where E is the number of directed edges, and the space complexity is $O(S^2)$.

Unreachability measure. One natural choice of deviation measure is the difficulty of reaching the baseline state s'_t from the current state s_t. Reachability of the starting state s_0 is commonly used as a constraint in reversibility-preserving approaches [Moldovan and Abbeel, 2012; Eysenbach *et al.*, 2017], where the agent does not take an action if it makes the reachability value function too low. The *unreachability (UR)* measure is defined as

$$d_{UR}(s_t; s'_t) := 1 - R(s_t; s'_t).$$

A problem with the unreachability measure is that it takes the maximum value of 1 if the agent takes any irreversible action (since the reachability of the baseline becomes 0). Thus, the agent receives the maximum penalty independently of the magnitude of the irreversible action, e.g. whether the agent breaks one vase or a hundred vases. This can lead to unsafe behavior, as demonstrated on the Box environment from the AI Safety Gridworlds suite [Leike *et al.*, 2017], shown in Figure 6.

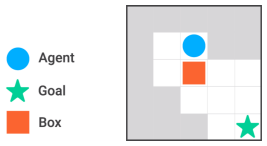


Figure 6: Box environment.

The environment contains a box that needs to be pushed out of the way for the agent to reach the goal. The unsafe behavior is taking the shortest path to the goal, which involves pushing the box down into a corner (an irrecoverable position). The desired behavior is to take a slightly longer path in order to push the box to the right. The action of moving the box is irreversible in both cases: if the box is moved to the right, the agent can move it back, but then the agent ends up on the other side of the box. Thus, the agent receives the maximum penalty of 1 for moving the box in any direction, so the penalty does not incentivize the agent to choose the safe path. Section 3 confirms that the unreachability penalty fails on the Box environment for all choices of baseline.

Relative reachability measure. To address the magnitude-sensitivity problem, we now introduce a reachability-based measure that is sensitive to the magnitude of the irreversible action. We define the *relative reachability (RR) measure* as the average reduction in reachability of all states s from the current state s_t compared to the baseline s'_t (Figure 1b):

$$d_{RR}(s_t; s'_t) := \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \max(R(s'_t; s) - R(s_t; s), 0)$$

The RR measure is nonnegative everywhere, and zero for states s_t that reach or exceed baseline reachability of all states.

In the Box environment, moving the box down makes more states unreachable than moving the box to the right (all states where the box is not in a corner become unreachable). Thus, the agent receives a higher penalty for moving the box down, and has an incentive to move the box to the right.

Modifications for the stepwise inaction baseline. In order to capture the delayed effects of actions, we modify the deviation measures to incorporate the inaction rollouts from s_t and $s'_t = s^{(t-1)}_t$ (shown in Figure 5) as follows:

$$d_{SUR}(s_t; s'_t) := 1 - (1 - \gamma) \sum_{k=0}^{\infty} \gamma^k R(s^{(t)}_{t+k}; s'^{(t)}_{t+k})$$

$$RV(\tilde{s}_t; s) := (1 - \gamma) \sum_{k=0}^{\infty} \gamma^k R(\tilde{s}^{(t)}_{t+k}; s)$$

$$d_{SRR}(s_t; s'_t) := \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \max(RV(s'_t; s) - RV(s_t; s), 0)$$

We call $RV(\tilde{s}_t; s)$ the *rollout value* of $\tilde{s}_t \in \{s_t, s'_t\}$ with respect to s. In a deterministic environment, the UR measure $d_{SUR}(s_t; s'_t)$ and the rollout value $RV(\tilde{s}_t; s)$ for the RR mea-

sure $d_{SRR}(s_t; s'_t)$ can be computed recursively as follows:

$$d_{SUR}(x; y) = (1-\gamma)(R(x; y) + \gamma d_{SUR}(I(x); I(y)))$$
$$RV(x; y) = (1-\gamma)(R(x; y) + \gamma RV(I(x); y))$$

where $I(x)$ is the inaction function that gives the state reached by following the inaction policy from state x (this is the identity function in static environments).

3 Experiments

We run a tabular Q-learning agent with different penalties on the gridworld environments introduced in Section 2 . While these environments are simple, they provide a proof of concept by clearly illustrating the desirable and undesirable behaviors, which would be more difficult to isolate in more complex environments. We compare all combinations of the following design choices for an impact penalty:

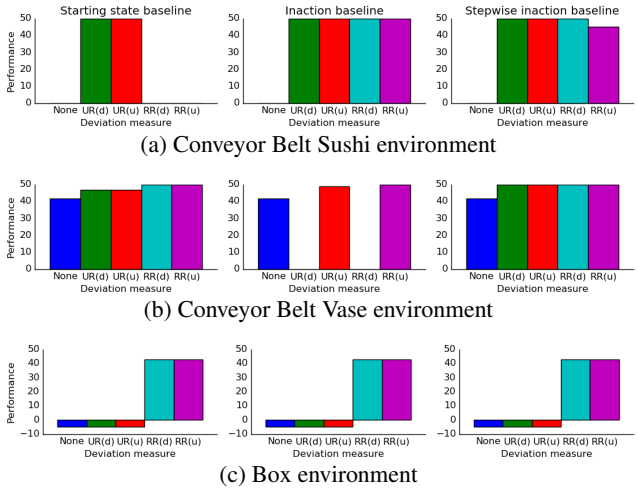
- Baselines: starting state s_0, inaction $s_t^{(0)}$, stepwise inaction $s_t^{(t-1)}$
- Deviation measures: unreachability (UR) ($d_{SUR}(s_t; s'_t)$ for the stepwise inaction baseline, $d_{UR}(s_t; s'_t)$ for other baselines), relative reachability (RR) ($d_{SRR}(s_t; s'_t)$ for the stepwise inaction baseline, $d_{RR}(s_t; s'_t)$ for other baselines), no penalty (None)
- Discounting: $\gamma_r = 0.99$ (discounted), $\gamma_r = 1.0$ (undiscounted)

In addition to the reward function, each environment has a *performance* function, originally introduced in Leike *et al.* [2017], which is not observed by the agent. This represents the agent's performance according to the designer's true preferences: it reflects how well the agent achieves the objective and whether it does so safely. In all environments, reaching the goal (if there is one) gives a reward of 50 and terminates the episode, and episode performance is obtained by subtracting 50 from the episode return if an unsafe outcome occurs. The Box environment has a movement penalty of -1, while the conveyor belt environments do not. The results are shown in Figure 7.

We anneal the exploration rate linearly from 1 to 0 over 9000 episodes, and keep it at 0 for the next 1000 episodes. For each penalty on each environment, we use a grid search to tune the scaling parameter β, choosing the value of β that gives the highest average performance on the last 100 episodes. The grid search is over $\beta = 0.1, 0.3, 1, 3, 10, 30, 100, 300, 1000$. We found that the agent performance was not very sensitive to the value of β, with near-optimal performance for a wide range of β values for each penalty.

Conveyor Belt Sushi environment (Figure 2). This environment has around 400 states. The safe outcome (reaching the goal without interfering with the sushi) achieves a performance of 50, while the unsafe outcome (taking the sushi off the belt) achieves a performance of 0.

An agent with no penalty achieves performance 0. It is unclear why it chooses an interference path, since non-interference paths have the same length and thus the same movement penalty. An agent with no penalty without annealing chooses interference path some of the time, resulting in a performance of 20, so its behavior is not stable.



(a) Conveyor Belt Sushi environment

(b) Conveyor Belt Vase environment

(c) Box environment

Figure 7: Performance results for different combinations of design choices (averaged over 10 seeds). The columns are different baseline choices: starting state, inaction, and stepwise inaction. The bars in each plot are results for different deviation measures (None, UR, and RR), with discounted and undiscounted versions indicated by (d) and (u) respectively.

All penalties with the inaction and stepwise inaction baselines reach near-optimal performance. The RR penalty with the starting state baseline produces the interference behavior (removing the sushi from the belt), resulting in performance 0. However, since the starting state is unreachable no matter what the agent does, the UR penalty is always at the maximum value of 1, so it does not produce interference behavior.

Conveyor Belt Vase environment (Figure 3). This environment has around 400 states. The agent receives a reward of 50 for taking the vase off the belt, which does not terminate the episode (it always lasts for 100 steps). The safe outcome (the vase is not broken) achieves performance 50, and the unsafe outcome (the vase is broken) achieves performance 0.

An agent with no penalty achieves performance 42. With the inaction baseline, the discounted penalties achieve performance 0, which corresponds to the offsetting behavior of moving the vase off the belt and then putting it back on, shown in Figure 4. The undiscounted versions avoid this behavior, since the action of taking the vase off the belt is reversible and thus is not penalized at all, so there is nothing to offset. All penalties with the stepwise inaction baseline perform well on this environment, showing that this baseline does not produce offsetting.

Box environment (Figure 6). This environment has 60 states. The safe longer path to the goal achieves performance 43, while the unsafe shorter path that puts the box in the corner achieves performance -5.

An agent with no penalty takes the unsafe path. For all baselines, RR achieves optimal performance 43, while UR achieves performance -5. This happens because the UR measure is not magnitude-sensitive, and thus does not distinguish between irreversible actions that result in recoverable and irrecoverable box positions.

Overall, the combinations of design choices that perform

best across all environments are RR with the stepwise inaction baseline and undiscounted RR with the inaction baseline. Since undiscounted RR only penalizes irreversible transitions, a penalty that aims to penalize reversible transitions has to be combined with the stepwise inaction baseline.

4 Additional related work

Safe exploration. Safe exploration methods prevent the agent from taking harmful actions by enforcing safety constraints [Turchetta *et al.*, 2016], penalizing risk [Chow *et al.*, 2015], using intrinsic motivation [Lipton *et al.*, 2016], preserving reversibility [Moldovan and Abbeel, 2012; Eysenbach *et al.*, 2017], etc. Explicitly defined constraints or safe regions tend to be task-specific and require significant human input, so they do not provide a general solution to the side effects problem. Penalizing risk and intrinsic motivation can help the agent avoid low-reward states (such as getting trapped or damaged), but do not discourage the agent from damaging the environment if this is not accounted for in the reward function. Reversibility-preserving methods have interference and magnitude insensitivity issues as discussed in Section 2.

Side effects criteria using state features. Zhang *et al.* [2018] assumes a factored MDP where the agent is allowed to change some of the features and proposes a criterion for querying the supervisor about changing other features in order to allow for intended effects. Shah *et al.* [2019] define an auxiliary reward for avoiding side effects in terms of state features by assuming that the starting state of the environment is already organized according to human preferences. Since the latter method uses the starting state as a baseline, we would expect it to produce interference behavior in dynamic environments. While these approaches are promising, they are not general in their present form due to reliance on state features.

Empowerment. Our RR measure is related to *empowerment* [Klyubin *et al.*, 2005; Salge *et al.*, 2014; Gregor *et al.*, 2017], a measure of the agent's control over its environment, defined as the highest possible mutual information between the agent's actions and the future state. Empowerment measures the agent's ability to reliably reach many states, while RR penalizes the reduction in reachability of states. Maximizing empowerment would encourage the agent to avoid irreversible side effects, but would also incentivize interference behavior, and it is unclear to us how to define an empowerment-based measure that would avoid this. One possibility is to penalize the reduction in empowerment between the current state s_t and the baseline s'_t. However, empowerment is indifferent between these two cases: A) the same states are reachable from s_t and s'_t, and B) a state x reachable from s'_t but unreachable from s_t, while another state y is reachable from s_t but unreachable from s'_t. Thus, penalizing reduction in empowerment would miss some side effects: e.g. if the agent replaced the sushi on the conveyor belt with a vase, empowerment could remain the same, so the agent is not penalized for destroying the vase.

Uncertainty about the objective. Inverse Reward Design [Hadfield-Menell *et al.*, 2017] incorporates uncertainty about the objective by considering alternative reward functions that are consistent with the given reward function in the training environment. This helps avoid some side effects that stem from distributional shift However, this method assumes that the given reward function is correct for the training environment, and so does not prevent side effects caused by a reward function that is misspecified in that environment. Quantilization [Taylor, 2016] incorporates uncertainty by taking actions from the top quantile of actions. These methods help to prevent side effects, but do not provide a way to quantify side effects.

Human oversight. An alternative to specifying a side effects penalty is to teach the agent to avoid side effects through human oversight, such as inverse reinforcement learning [Ng and Russell, 2000; Hadfield-Menell *et al.*, 2016], demonstrations [Abbeel and Ng, 2004], or human feedback [Christiano *et al.*, 2017; Saunders *et al.*, 2017]. It is unclear how well an agent can learn a general heuristic for avoiding side effects from human oversight. We expect this to depend on the diversity of settings in which it receives oversight and its ability to generalize from those settings, which are difficult to quantify. We expect that an intrinsic penalty for side effects would be more robust and more reliably result in avoiding them. Such a penalty could also be combined with human oversight to decrease the amount of human input required for an agent to learn human preferences.

5 Conclusions

We have outlined possible bad incentives (interference and offsetting) that can arise from a poor choice of baseline, and a magnitude insensitivity property that can arise from a poor choice of deviation measure, and proposed design choices that avoid these issues in preliminary experiments. There are many possible directions for follow-up work:

Scalable implementation. The RR measure in its exact form is not tractable for environments more complex than gridworlds. A more practical implementation could be computed over some set of representative states instead of all states. For example, the agent could learn a set of auxiliary policies for reaching distinct states, similarly to the method for approximating empowerment in Gregor *et al.* [2017].

Better choices of baseline. Using noop actions to define inaction for the stepwise inaction baseline can be problematic. For example, when driving a car on a winding road, the default outcome of a noop is a crash, so the agent would not be penalized for spilling coffee in the car. This could be avoided using a better inaction baseline, such as following the road, but this can be challenging to define in a task-independent way.

Weights over the state space. In practice, we often value the reachability of some states much more than others. This could be incorporated into the relative reachability measure by adding a weight w_s for each state s in the sum. Such weights could be learned through human feedback methods, e.g. Christiano *et al.* [2017].

Theory. There is a need for theoretical work on characterizing and formalizing undesirable incentives that arise from different design choices in penalizing side effects.

We hope this work lays the foundations for a practical methodology on avoiding side effects that would scale well to more complex environments.

References

Pieter Abbeel and Andrew Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, pages 1–8, 2004.

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

Stuart Armstrong and Benjamin Levinstein. Low impact artificial intelligences. *arXiv preprint arXiv:1705.10720*, 2017.

Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a CVaR optimization approach. In *Neural Information Processing Systems (NIPS)*, pages 1522–1530, 2015.

Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Neural Information Processing Systems (NIPS)*, 2017.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no Trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

Jaime F. Fisac, Anayo K. Akametalu, Melanie Nicole Zeilinger, Shahab Kaynama, Jeremy H. Gillula, and Claire J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *arXiv preprint arXiv:1705.01292*, 2017.

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

Jeremy H. Gillula and Claire J. Tomlin. Guaranteed safe online learning via reachability: tracking a ground target using a quadrotor. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2723–2730, 2012.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *International Conference for Learning Representations (ICLR) Workshop, arXiv preprint arXiv:1611.07507*, 2017.

Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, and Stuart Russell. Cooperative inverse reinforcement learning. In *Neural Information Processing Systems (NIPS)*, 2016.

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J. Russell, and Anca D. Dragan. Inverse reward design. In *Neural Information Processing Systems (NIPS)*, pages 6768–6777, 2017.

Alexander S. Klyubin, Daniel Polani, and Chrystopher L. Nehaniv. All else being equal be empowered. In *European Conference on Artificial Life (ECAL)*, pages 744–753, 2005.

Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A. Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. AI safety gridworlds. *arXiv preprint arXiv:1711.09883*, 2017.

Zachary C. Lipton, Jianfeng Gao, Lihong Li, Jianshu Chen, and Li Deng. Combating reinforcement learning's sisyphean curse with intrinsic fear. *arXiv preprint arXiv:1611.01211*, 2016.

John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, pages 463–502. Edinburgh University Press, 1969.

Ian M. Mitchell, Alexandre M. Bayen, and Claire J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, 2005.

Teodor Mihai Moldovan and Pieter Abbeel. Safe exploration in Markov decision processes. In *International Conference on Machine Learning (ICML)*, pages 1451–1458, 2012.

Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, pages 663–670, 2000.

Pedro Ortega, Vishal Maini, and et al. Building safe artificial intelligence: specification, robustness, and assurance. DeepMind Safety Research Blog, 2018.

Martin Pecka and Tomas Svoboda. Safe exploration techniques for reinforcement learning — an overview. In *International Workshop on Modelling and Simulation for Autonomous Systems*, pages 357–375, 2014.

Christoph Salge, Cornelius Glackin, and Daniel Polani. Empowerment — an introduction. In *Guided Self-Organization: Inception*, pages 67–114. Springer, 2014.

William Saunders, Girish Sastry, Andreas Stuhlmueller, and Owain Evans. Trial without error: Towards safe reinforcement learning via human intervention. *arXiv preprint arXiv:1707.05173*, 2017.

Rohin Shah, Dmitrii Krasheninnikov, Jordan Alexander, Pieter Abbeel, and Anca Dragan. Preferences implicit in the state of the world. In *International Conference for Learning Representations (ICLR)*, 2019.

Jessica Taylor, Eliezer Yudkowsky, Patrick LaVictoire, and Andrew Critch. Alignment for advanced machine learning systems. Technical report, Machine Intelligence Research Institute, 2016.

Jessica Taylor. Quantilizers: A safer alternative to maximizers for limited optimization. In *AAAI Workshop on AI, Ethics, and Society*, pages 1–9, 2016.

Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration in finite Markov decision processes with Gaussian processes. In *Neural Information Processing Systems (NIPS)*, pages 4305–4313, 2016.

Alexander Turner, Dylan Hadfield-Menell, and Prasad Tadepalli. Conservative agency via attainable utility preservation. *arXiv preprint arXiv:1902.09725*, 2019.

Shun Zhang, Edmund H. Durfee, and Satinder P. Singh. Minimax-regret querying on side effects for safe optimality in factored Markov decision processes. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4867–4873, 2018.