Command-Space Counterfactual Explanations for Pareto-Conditioned Reinforcement Learning

Joanikij Chulev[1] and **Hendrik Baier**[1,2]

[1] Centrum Wiskunde & Informatica (CWI), Amsterdam

[2]Eindhoven University of Technology

{joanikij.chulev, hendrik.baier}@cwi.nl

Abstract

Pareto Conditioned Networks learn multiple multi-objective reinforcement learning behaviours by conditioning a single policy on a desired return command. However, the local mapping from command and state to action remains opaque. We propose command-space counterfactual explanations for PCNs: given a fixed state, original command, and foil action, we search, in a black-box setting, for a minimally changed desired-return command under which the same trained policy would choose the foil. Our contributions are threefold. First, we formulate PCN explanations as return-command interventions, using a return-only PCN variant that avoids the added ambiguity of horizon-conditioning. Second, we adapt adversarial machine learning methods to reinforcement-learning explanations. Third, we introduce a boundary-seeded directional search that improves over purely local optimization in the command-action landscape, resulting in our proposed approach CF-ZOO. The resulting explanations are actionable and intuitively expressed in the user's own preferences: "If your trade-off had shifted slightly towards X, the agent would have chosen Y."

1 Introduction

Many sequential decision problems are inherently multi-objective: an agent may need to trade off aspects such as safety, efficiency or cost rather than optimize a single scalar reward. Multi-objective reinforcement learning (MORL) addresses this setting by using vector-valued rewards and learning policies that represent trade-offs among objectives [Roijers *et al.*, 2013; Hayes *et al.*, 2022]. Such objective structure is also useful for XRL, where MORL has been identified as a route to contrastive explanation through explicit trade-offs among objectives [Milani *et al.*, 2024; Dazeley *et al.*, 2023]. The desired output is not one optimal policy but a set of Pareto-efficient behaviours, where improving one objective requires sacrificing another [Roijers *et al.*, 2013].

Pareto Conditioned Networks (PCNs) provide a compact way to represent such behaviours. Instead of learning a separate policy for each trade-off, a PCN learns one neural policy conditioned on a desired return and horizon [Reymond *et al.*, 2022]. At deployment, different desired-return commands can induce different behaviours from the same trained network. This makes PCNs attractive for decision support, but it does not make their decisions transparent: the command is interpretable, while the local mapping from state and command to action remains a neural black box.

Counterfactual explanations are well suited to this gap. In supervised learning, they explain how an input would need to change for a model to produce a different output [Wachter *et al.*, 2018; Verma *et al.*, 2024]. In RL, however, counterfactuals are more complex, because decisions are sequential, stochastic, and connected to goals, plans, and future outcomes [Gajcin and Dusparic, 2024c]. Existing RL counterfactual work has therefore focused on visual state counterfactuals [Huber *et al.*, 2023; Samadi *et al.*, 2024], reachable and certain state counterfactuals [Gajcin and Dusparic, 2024b], diverse counterfactual action sequences [Gajcin and Dusparic, 2024a], and counterfactual modifications to policies themselves [Deshmukh *et al.*, 2023].

These approaches do not directly target the distinctive explanatory interface of a PCN: the desired-return command. For a PCN, a natural local question is: given the same state and the same trained policy, what minimal change to the command would make a difference? We propose an explanation, contrastive at the action level and interpretable at the objective level: it answers what desired-return trade-off would have made the same policy choose a different action in the same state. The result is directly actionable for the user. To compute these explanations, we propose CF-ZOO.

Methodologically, CF-ZOO adapts tools from adversarial and black-box optimization. Carlini–Wagner attacks formulate targeted behavioural change as an optimization problem balancing input proximity against a target-class loss [Carlini and Wagner, 2017]. ZOO replaces back-propagation with finite-difference zeroth-order queries in black-box settings [Chen *et al.*, 2017]. Cheng et al. instead search over directions and scalar distances to decision boundaries to locate small hard-label perturbations [Cheng *et al.*, 2019]. Related work shows that adversarial perturbations can affect RL policies in domains such as algorithmic trading and autonomous lane changing [Piazza *et al.*, 2021; Zhang *et al.*, 2023]. We repurpose this optimization logic for explanation rather than attack: the perturbed object is the PCN command, and the

goal is to expose how local action choice depends on the requested multi-objective trade-off.

A simple example illustrates the kind of explanation we seek. In Deep Sea Treasure, a submarine trades off treasure value against time cost. At one state, the submarine is one step above a nearby treasure of value 5, while moving right continues toward a larger treasure of value 8. The first component of the reward is the desired treasure value, and the second component is the desired time-cost return. Under the remaining command $R_t = (8.000, -2.000)$, the trained PCN moves right. A user may ask: what would need to change for the same agent, in the same state, to move down instead? Our method returns $R_{\text{cf}} = (6.328, -1.704)$, with $\delta R = (-1.672, 0.296)$. This means that if the desired treasure return were reduced and the command placed slightly more pressure on finishing sooner, the same policy would move down to collect the nearer treasure. The explanation therefore reveals what the PCN has learned locally: the original action is tied to the command's preference for the larger treasure, not only to the submarine's current position. The user thus learns how to steer the policy toward the value-5 treasure as well.

Code is available at: https://github.com/JoanikijChulev/Counterfactual-Explanations-for-Pareto-Conditioned-RL.

2 Background

2.1 Multi-Objective Reinforcement Learning

A standard reinforcement learning problem is commonly formalized as a Markov decision process (MDP), where an agent observes a state, selects an action, receives a reward, and transitions to a next state [Sutton and Barto, 2018]. MORL generalizes this setting by replacing the scalar reward with a vector reward. A multi-objective MDP can be written as

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, \mathbf{r}, \gamma \rangle,$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s' \mid s, a)$ is the transition function, $\mathbf{r}(s, a, s') \in \mathbb{R}^m$ is an m-dimensional reward vector, and $\gamma \in [0, 1]$ is a discount factor [Roijers *et al.*, 2013; Hayes *et al.*, 2022]. For a policy π, the return is therefore vector-valued:

$$\mathbf{G}_t = \sum_{k=t}^{T} \gamma^{k-t} \mathbf{r}_k.$$

MORL usually reasons in terms of Pareto dominance. A return vector $\mathbf{x}$ dominates $\mathbf{y}$ if it is at least as good in every objective and strictly better in at least one.

One way to solve a MORL problem is scalarization, where a utility function $u : \mathbb{R}^m \to \mathbb{R}$ maps vector returns to a scalar objective. However, scalarization requires preferences to be specified and can miss parts of the Pareto front, especially under linear utilities [Roijers *et al.*, 2013; Hayes *et al.*, 2022]. Multi-policy and conditioned-policy methods instead aim to represent many trade-offs within one learned system. PCNs are one such conditioned-policy approach. Related approaches include preference-conditioned methods such as PD-MORL, which trains a single universal policy over preference space [Basaklar *et al.*, 2023].

2.2 Pareto Conditioned Networks and Our Return-Only Variant

PCNs learn a single policy network conditioned on a desired outcome [Reymond *et al.*, 2022]. In the original formulation, the policy is conditioned on the current state, a desired return vector, and a desired horizon:

$$\pi_\theta(a_t \mid s_t, R_t^{\text{des}}, h_t^{\text{des}}), \tag{1}$$

where R_t^{des} specifies the desired multi-objective return and h_t^{des} specifies the desired number of steps. During training, PCN stores experienced transitions together with their achieved returns and horizons, and learns to reproduce actions under the corresponding return–horizon commands [Reymond *et al.*, 2022]. Although horizon is part of the original PCN command, the original paper does not specifically show its role experimentally.

In this paper, we use a revised return-only variant:

$$\pi_\theta(a_t \mid s_t, R_t^{\text{des}}). \tag{2}$$

During testing, we observed that horizon conditioning was not always a reliable control signal. Even when the desired return command corresponded to a Pareto-optimal point, changing only the desired horizon could sometimes produce unstable or unintuitive action choices. This suggests that the horizon input may not have consistently encoded useful temporal guidance for the policy. Using the return-only formulation, we trained PCN agents and recovered the expected Pareto-optimal trade-off structure in our environments. For example, the learned Minecart environment front contained more Pareto-optimal solutions than reported for the original implementation [Reymond *et al.*, 2022; Abels *et al.*, 2019]. Figure 1 shows the Pareto front recovered by our return-only variant on Minecart.

Furthermore, our explanation objective is to understand how desired-return commands affect action choice. Keeping the horizon would require counterfactuals over both return and time,

$$(R, h) \mapsto (R + \delta_R, h + \delta_h),$$

whereas the return-only formulation yields the simpler explanation problem

$$R \mapsto R + \delta_R.$$

For a fixed state s, the relevant object is therefore the local action distribution induced by the return command. Let $\pi_\theta(s, R) \in \mathbb{R}^{|\mathcal{A}|}$ denote the action probabilities produced by the policy. The greedy action is

$$a^\star(s, R) = \arg\max_{a \in \mathcal{A}} \pi_\theta(a \mid s, R). \tag{3}$$

A command-space counterfactual then asks for a small perturbation δ_R such that the preferred action changes. This makes the desired return an interpretable intervention point.

2.3 Counterfactual Explanations in Reinforcement Learning

A counterfactual explanation is usually defined relative to an original input, an original model output, and a desired alternative output [Wachter *et al.*, 2018; Verma *et al.*, 2024].

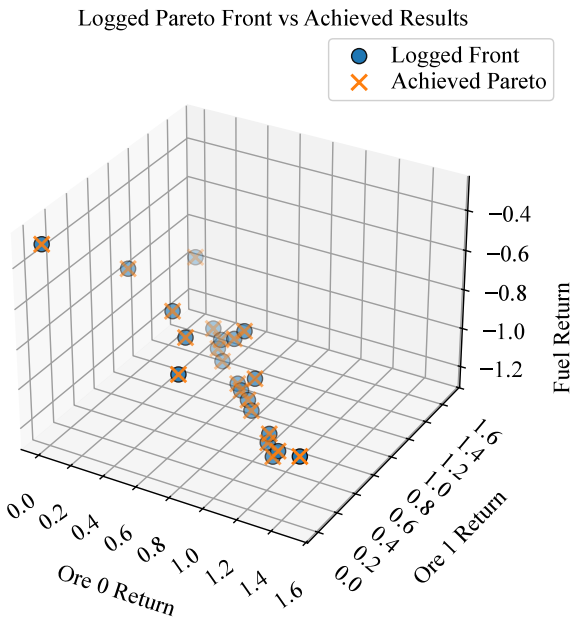

Figure 1: Pareto front recovered by the return-only PCN variant on Minecart. Blue points (O) show the final logged learning front; orange points (X) show rollout-achieved returns.

In supervised learning, counterfactual search is often formulated as an optimization problem combining validity, proximity, sparsity, plausibility, and sometimes diversity [Mothilal *et al.*, 2020; Dandl *et al.*, 2020].

In RL, these desiderata are not sufficient. RL decisions occur inside temporally extended interaction loops, so a counterfactual state that is close in feature space can still be unreachable from the original state under the environment dynamics [Gajcin and Dusparic, 2024c]. Stochasticity adds another complication: a counterfactual may produce the desired action or outcome in one rollout but fail under nearby random transitions.

Existing work addresses these issues in several ways. Olson et al. generate counterfactual state explanations for visual RL agents using generative models [Olson *et al.*, 2021]. Tsirtsis et al. formulate counterfactual explanations for sequential decision making under uncertainty as alternative action sequences or policies [Tsirtsis *et al.*, 2021]. RACCER introduces RL-specific properties such as reachability, stochastic certainty, and fidelity [Gajcin and Dusparic, 2024b], while ACTER extends the focus to diverse counterfactual action sequences [Gajcin and Dusparic, 2024a]. COViz explains local RL decisions by comparing the outcome of the chosen action with a counterfactual action outcome [Amitai *et al.*, 2024].

GANterfactual-RL and SAFE-RL focus on visual counterfactuals for deep RL policies, where the goal is to alter high-dimensional observations in ways that change the agent's action while preserving plausibility [Huber *et al.*, 2023; Samadi *et al.*, 2024].

PCN command-space counterfactuals differ from these approaches. They do not modify pixels, symbolic state variables, or past action sequences. They ask how the user's requested trade-off would need to change for a different local action to become preferred. Such explanations are especially natural for MORL because the focus is on objectives.

3 Command-Space Counterfactual Search

Let s_t be the state to explain, $R_t \in \mathbb{R}^m$ the remaining desired-return command, $a^\star$ the greedy action selected by the PCN, and $a_f \neq a^\star$ a valid foil action. The policy is queried only as a black box: for a command R, it returns log-probabilities $\ell_a(s_t, R)$. The valid action set at s_t is denoted by $\mathcal{A}(s_t)$.

A command-space counterfactual is a command

$$R_{cf} = R_t + \delta \tag{4}$$

such that the foil becomes the greedy action:

$$a_f = \arg\max_{a \in \mathcal{A}(s_t)} \ell_a(s_t, R_{cf}). \tag{5}$$

The resulting explanation has the form: the policy selected $a^\star$ under R_t, but would have selected a_f under R_{cf}. The state and policy are held fixed; only the desired-return command is changed.

3.1 Optimization Objective

Our loss adapts the targeted margin structure used in Carlini–Wagner-style attacks and ZOO [Carlini and Wagner, 2017; Chen *et al.*, 2017]. The difference is semantic: adversarial attacks perturb an input to induce misclassification, whereas we perturb the PCN command to expose how the local action preference depends on the requested multi-objective return.

For a foil a_f, define the target margin:

$$M_f(R) = \ell_{a_f}(s_t, R) - \max_{a \in \mathcal{A}(s_t),\, a \neq a_f} \ell_a(s_t, R). \tag{6}$$

The foil is greedy when $M_f(R) \geq 0$. We use a margin parameter $\kappa \geq 0$ and require $M_f(R) \geq \kappa$, so that the foil wins by a nonzero margin when strict flips are desired. Equivalently, define the hinge term

$$H_f(R) = \max\{\kappa - M_f(R), 0\}. \tag{7}$$

This term is zero exactly when the foil action beats all other valid actions by at least κ.

The black-box objective optimized by the explainer is

$$\mathcal{L}(\delta) = \|\delta\|_2^2 + c\, H_f(\Pi_{\mathcal{C}}(R_t + \delta)), \tag{8}$$

where $c > 0$ controls the strength of the targeted flip penalty and $\Pi_{\mathcal{C}}$ clips commands to the feasible box. The first term selects small command changes. The second term enforces the foil-action preference.

3.2 Boundary-Seeded Directional Search

A purely local C&W/ZOO-style search can fail when the command-action landscape is flat, non-convex, or when the current command is far from the foil region. In that case, finite-difference updates around R_t may only observe weak local changes in the target margin and can converge to an uninformative local solution that never makes the foil action greedy. We therefore add directional boundary seeding. The purpose is not to replace the C&W/ZOO objective, but to provide it with a better initial perturbation direction.

This idea is adapted from query-efficient hard-label black-box attacks. Cheng et al. reformulate hard-label attack search by optimizing over directions rather than directly over perturbed inputs [Cheng *et al.*, 2019]. For an input x_0, classifier f, and target class t, their targeted boundary-distance objective is

$$g(\theta) = \min_{\lambda>0} \lambda \quad \text{s.t.} \quad f\left(x_0 + \lambda \frac{\theta}{\|\theta\|_2}\right) = t. \tag{9}$$

Inspired by these attacks, we use a directional command-space search: candidate directions are evaluated by testing their feasible endpoint, and successful directed perturbations are binary-refined back toward the original command to approximate the nearest successful perturbation along that direction.

The seeded directional search assumes access to the training-time Pareto archive. Thus, we check for counterfactuals in the directions of all found Pareto front solutions, relying on them as behavioral heuristics. Let $\mathcal{F} = \{R_j^{\text{front}}\}_{j=1}^{N}$ denote the logged Pareto-front returns. At timestep t, the current command R_t is the remaining part of the originally selected command R_{des}. Hence the return already collected is

$$R_t^{\text{col}} = R_{\text{des}} - R_t.$$

Each front point is converted into the remaining command that would still be needed to reach it:

$$\tilde{R}_j = R_j^{\text{front}} - R_t^{\text{col}}.$$

We query the PCN at each $\tilde{R}_j$, compute the foil margin $M_f(\tilde{R}_j)$, and select

$$j^\star = \arg\max_j M_f(\tilde{R}_j).$$

This candidate is used only to infer a coarse direction of useful command change. Specifically, the directional sign is computed from

$$d = \tilde{R}_{j^\star} - R_t.$$

For component i, if $d_i > 0$, subsequent ray search is restricted to increasing that command component; if $d_i < 0$, it is restricted to decreasing that component; and if $d_i = 0$, no directional restriction is imposed on that component.

After inferring the directional prior, we sample unit candidate directions as

$$v \sim \mathcal{N}(0, I_d), \qquad u = \frac{v}{\|v\|_2}, \tag{10}$$

where I_d is the $d \times d$ identity matrix, so v is a d-dimensional standard Gaussian vector. For each u, we search along the command-space ray, where $D = \text{diag}(R_{\max} - R_{\min})$ scales the command dimensions

$$R(\alpha; u) = R_t + \alpha D u, \qquad \alpha \geq 0. \tag{11}$$

The endpoint of the perturbation is the largest feasible α that remains inside both the reward bounds and the directional bounds inferred from the Pareto-front heuristic. If this endpoint does not satisfy

$$M_f(R(\alpha; u)) \geq \kappa, \tag{12}$$

the point is discarded. If the endpoint succeeds, we binary-search back toward R_t to remove unnecessary perturbation magnitude. All successful candidates are retained and compared using the scaled command distance

$$d_D(R, R_t) = \left\| D^{-1}(R - R_t) \right\|_2. \tag{13}$$

The closest successful ray candidate becomes the seed for the subsequent ZOO coordinate refinement. If no ray succeeds, the method falls back to local ZOO from the original command. Thus, the boundary-seeding phase supplies a global directional guess, while the ZOO phase performs local black-box refinement around the best found candidate.

3.3 ZOO Perturbation Refinement

After directional seeding, we refine the best seed using ZOO-style zeroth-order coordinate optimization [Chen *et al.*, 2017]. ZOO replaces back-propagation with finite-difference queries to the target model. In our setting, the target model is the PCN queried at the fixed state s_t, and the optimized variable is the command perturbation δ.

At each iteration, a coordinate $i \in \{1, \ldots, d\}$ is sampled. The coordinate derivative of Eq. (8) is estimated by central finite differences:

$$\widehat{\nabla_i \mathcal{L}}(\delta) = \frac{\mathcal{L}(\delta + he_i) - \mathcal{L}(\delta - he_i)}{2h}, \tag{14}$$

where e_i is the i-th basis vector and h is the finite-difference step. Each coordinate update therefore uses two score queries for the finite-difference estimate and one additional query to evaluate the updated command. We use ZOO-ADAM for optimization of coordinate updates.

4 Worked Examples

Figure 2 shows the three environments used for qualitative inspection. The purpose is not only to show that an action flip occurs, but also to show how the result can be communicated as a human-facing explanation of the agent's behaviour.

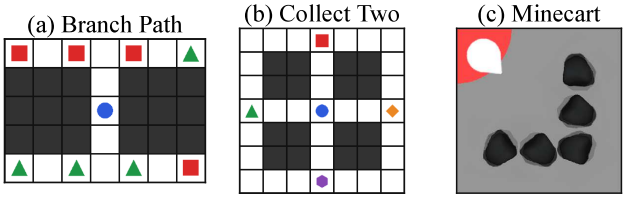

Figure 2: (a) Branch-Path Environment - Custom. (b) Collect-Two Environment - Custom. (c) Minecart Environment - Classic.

4.1 Branch-Path

Branch-Path is a custom 5×7 two-objective grid with a central start state and two narrow branches. Collectibles of type A (red square) give reward $(1, 0)$, while collectibles of type B (green triangle) give reward $(0, 1)$. Invalid moves are masked, so the search only compares feasible actions. There are three Pareto-front points, due to an episode step limit of 13 steps. We select

$$R_t = (3, 1),$$

which prioritizes objective A. At timestep $t = 1$, the remaining command is still $R_t = (3, 1)$. The valid actions are `up` and `down`. The PCN assigns probability 0.995 to `up` and 0.005 to `down`, so

$$a^\star = \texttt{up}.$$

We choose the foil

$$a_f = \texttt{down}.$$

The search returns

$$R_{cf} = (0.2953, 1.464), \qquad \delta_R = (-2.7047, 0.464).$$

Under R_{cf}, the greedy action changes from `up` to `down`. The intuitive explanation here is: *At this state, the agent goes up because the current command strongly asks for objective A. If the user instead asked for much less A and somewhat more B, the same agent in the same state would go down. The upward action is therefore not arbitrary: it is tied to the command's preference for A.*

4.2 Collect-Two

Collect-Two is a custom 7×7 four-objective grid. The agent starts at the center, with objective A (red) above, B (green) left, C (yellow) right, and D (purple) below. The task terminates after two objectives are collected. The first collected objective gives reward 1.0, and the second gives reward 0.8. Thus, the Pareto front contains 12 points. At timestep $t = 0$, the agent is at the center state with command

$$R_t = (0, 0, 0.8, 1).$$

This command values objective C, but values objective D most. The PCN assigns probabilities 0.145, 0.128, 0.132, and 0.595 to `right`, `up`, `left`, and `down`, respectively. Therefore,

$$a^\star = \texttt{down}.$$

We choose

$$a_f = \texttt{right}.$$

The search returns

$$R_{cf} = (0, 0, 0.8, 0.8437), \qquad \delta_R = (0, 0, 0, -0.1563).$$

Under R_{cf}, we are making `right` greedy. We can explain this as: *At the center, the agent goes down because objective D is requested slightly more strongly than objective C. If the desired return for D were reduced and made similar to C, the agent would instead go right toward C. If the agent values them both similarly, it would go right. This is a sparse explanation: only one objective in the command needs to change.*

4.3 Minecart

Minecart is a continuous-state MORL benchmark in which a cart moves through a two-dimensional map, collects ore, and trades off ore objectives against fuel consumption [Abels *et al.*, 2019]. This example tests whether command counterfactuals remain interpretable outside small discrete grids.

At timestep $t = 5$, the state is

$$s_t = [0.4876, 0.1991, 0.03, 0.0872, 0.9962, 0, 0],$$

corresponding to a cart near $(x, y) = (0.488, 0.199)$, moving slowly, approximately facing east, with empty cargo. The command is

$$R_t = (0.28, 1.22, -0.96),$$

where the first two dimensions are ore objectives and the third is fuel-related. The PCN assigns probabilities 0.999860, 0.000005, and 0.000134 to `Left`, `Right`, and `None`, respectively, so

$$a^\star = \texttt{Left}.$$

We choose

$$a_f = \texttt{Right}.$$

The search returns

$$R_{cf} = (0.3476, 1.22, -0.96), \qquad \delta_R = (0.0676, 0, 0).$$

Under R_{cf}, `Right` becomes greedy, which is consistent with a sensible learned trade-off. This can be explained as: *Given the state, the agent strongly turns left under the original command. If the user requested slightly more of the first ore objective, while leaving the second ore objective and fuel command unchanged, the same agent would instead turn right. Increasing the desired amount of ore 0 makes the policy stop steering toward the nearby mine containing only ore 1 and instead steer clockwise toward mines that contain ore 0.*

5 Experimental Results

We use a mixture of custom diagnostic environments and standard MORL benchmarks, several of which are provided through MO-Gymnasium and MORL-Baselines [Felten *et al.*, 2023]. Deep Sea Treasure (DST) is a classic MORL grid benchmark in which a submarine trades off treasure value against a time penalty [Vamplew *et al.*, 2011]. Breakable-Bottles is a low-impact MORL benchmark where the agent must deliver bottles while accounting for time and potential environmental side effects [Vamplew *et al.*, 2021]. Resource-Gathering is a grid-world task in which the agent collects resources such as gold and gems while facing enemy risk, originally introduced by Barrett and Narayanan [Barrett and Narayanan, 2008]. Fruit-Tree Navigation is a tree-structured MORL benchmark in which each path leads to a fruit with a multi-objective nutrient vector, testing generalization over larger reward spaces [Yang *et al.*, 2019]. Reward-Line is our custom diagnostic environment: the agent moves in a grid to a terminal row where the terminal column defines a linear two-objective trade-off between $(1, 0)$ and $(0, 1)$.

5.1 White-Box Baseline

We first test whether the command counterfactual problem can be solved by a strong local optimizer when model internals are available. The baseline, denoted WHITE-CW, adapts the Carlini–Wagner targeted attack objective to PCN command vectors and optimizes it through the Adversarial Robustness Toolbox (ART) [Carlini and Wagner, 2017; Nicolae *et al.*, 2018]. Unlike our method, this baseline has white-box access to the model and can use internal gradients. A comparison to this baseline allows us to test how many of the failures are due to black-box access.

Our method, denoted CF-ZOO, uses the same foil-validity condition but first performs boundary-seeded directional

search before zeroth-order local refinement. A case is successful if the method finds a feasible command R_{cf} for which the selected foil action becomes greedy.

Env.	Method	Cases	Succ.	Rate [95% CI]	Dist.
Branch	WHITE-CW	35	7	20.0 [10.0,35.9]	0.270
Branch	CF-ZOO	35	25	71.4 [54.9,83.7]	0.416
B-Bottles	WHITE-CW	8	2	25.0 [7.1,59.1]	0.017
B-Bottles	CF-ZOO	8	6	75.0 [40.9,92.9]	0.030
Collect	WHITE-CW	136	29	21.3 [15.3,28.9]	0.514
Collect	CF-ZOO	136	136	100.0 [97.3,100.0]	0.444
DST	WHITE-CW	136	14	10.3 [6.2,16.5]	0.014
DST	CF-ZOO	136	50	36.8 [29.1,45.1]	0.152
Fruit	WHITE-CW	84	83	98.8 [93.6,99.8]	0.130
Fruit	CF-ZOO	84	84	100.0 [95.6,100.0]	0.121
Minecart	WHITE-CW	240	116	48.3 [42.1,54.6]	0.098
Minecart	CF-ZOO	240	208	86.7 [81.8,90.4]	0.153
R-Gather	WHITE-CW	10	0	0.0 [0.0,27.8]	–
R-Gather	CF-ZOO	10	4	40.0 [16.8,68.7]	1.186
R-Line	WHITE-CW	148	71	48.0 [40.1,56.0]	0.380
R-Line	CF-ZOO	148	124	83.8 [77.0,88.9]	0.363
All	WHITE-CW	797	322	40.4 [37.0,43.8]	–
All	CF-ZOO	797	637	79.9 [77.0,82.6]	–

Table 1: Comparison of our perturbation method with a white-box C&W baseline. Success-rate intervals are Wilson score 95% confidence intervals. Dist. denotes the mean scaled ℓ_2 distance over successful counterfactuals.

CF-ZOO succeeds in $637/797$ cases (79.9%), compared with $322/797$ (40.4%) for WHITE-CW. Except in Fruit-Tree, boundary seeding substantially improves validity, indicating that white-box access alone does not overcome poor local initialization. Distances are averaged only over successful cases, so larger CF-ZOO distances can reflect additional, harder cases that WHITE-CW does not solve. For example, in DST it succeeds in only $14/136$ cases with mean distance 0.014, whereas CF-ZOO succeeds in $50/136$ cases with mean distance 0.152. The larger distance for CF-ZOO reflects that it reaches foil regions that the local white-box optimizer misses. These results support the boundary-seeding design: even with model internals available, local C&W-style optimization often fails to cross the action boundary, while CF-ZOO more reliably reaches feasible foil regions.

5.2 Command-Space Landscape Exploration

A closer look at the command space helps explain why the two methods behave differently. Figure 3 shows four representative two-dimensional slices of the command space. In panels (a), (c), and (d), WHITE-CW finds no valid counterfactual, so its marker overlaps the original command R_t; only in panel (b) does it reach the validity margin. The arrows show the local direction of increasing foil margin and are diagnostic. In panel (a), this direction is poorly aligned with a nearby valid region, while panel (c) shows that even an apparently useful local direction does not guarantee success. These landscape figures therefore illustrate how such landscapes can limit purely local optimization. By screening nonlocal rays, CF-ZOO reaches a foil-valid region in all four cases.

5.3 Combinatorial Stress Test

We further evaluate our implementation in a stress test. Instead of selecting a small set of representative counterfactual queries, we generate explanation cases across available Pareto-front commands, sampled timesteps, and all valid foil actions at each selected state. This produces a full evaluation of whether the method performs under many decision contexts. The white-box comparison (Table 1) used the same procedure restricted to fewer front commands, early timesteps and fewer foils so its cases are a subset of Table 2.

Environment	Cases	Succ.	Rate [95% CI]	Dist.
Branch-Path	44	30	68.2 [53.4,80.0]	0.466
B-Bottles	8	6	75.0 [40.9,92.9]	0.030
Collect-Two	152	152	100.0 [97.5,100.0]	0.493
DST	243	70	28.8 [23.5,34.8]	0.641
Fruit-Tree	672	672	100.0 [99.4,100.0]	0.231
Minecart	1846	1341	72.6 [70.6,74.6]	0.360
R-Gather	29	10	34.5 [19.9,52.7]	1.131
Reward-Line	201	141	70.1 [63.5,76.0]	0.525
Total	3195	2422	75.8 [74.3,77.3]	–

Table 2: Stress test of CF-ZOO. Each case corresponds to a state, command, and valid foil action. Success-rate intervals are Wilson score 95% confidence intervals. Dist. denotes the mean scaled ℓ_2 distance over successful counterfactuals.

Across $3{,}195$ cases, CF-ZOO succeeds in $2{,}422$ (75.8%). The method succeeds on all Collect-Two and Fruit-Tree cases, and achieves high success on Minecart and Reward-Line, indicating that the search scales beyond small diagnostic grids and remains effective in continuous-state and structured multi-objective settings. The lower success rates in DST and Resource-Gathering suggest that some environments contain local decisions where the foil action is difficult to induce through command changes alone. In these cases, the policy may be strongly constrained by the state or the desired foil may lie outside the learned behavioural support. The mean scaled distances show that the magnitude of successful explanations varies substantially across environments. Overall, the stress test shows that boundary-seeded command search can recover counterfactual explanations for most generated foil queries while also exposing where the PCN does not provide reliable command-level control.

To determine whether these failures reflect a limitation of the search or the absence of a counterfactual among the evaluated commands, we exhaustively evaluated a dense finite grid over the command space at each failing case. A foil for which no grid command flips the action is treated as heuristic evidence that the policy may not have learned that behaviour. Table 3 decomposes CF-ZOO's outcomes. Of all cases, 78.7% are solved, while 16.6% are grid-infeasible foils—actions that no evaluated grid command flips. Restricting to the known-feasible cases, for which CF-ZOO or the grid found at least one valid counterfactual, CF-ZOO recovers 94.4%. The remaining 5.6% are search failures that persist even when the search budget is high, indicating a small set of known-feasible counterfactuals that the method struggles to locate.

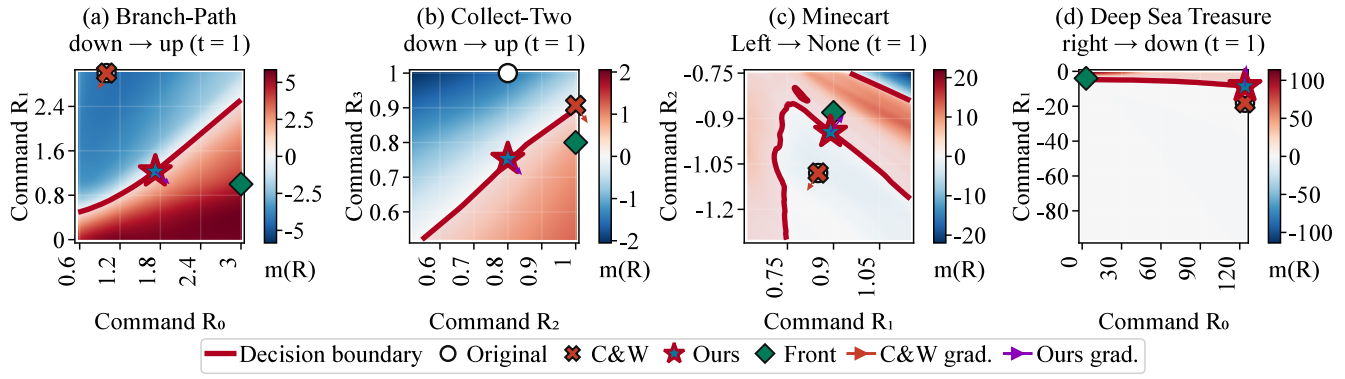

Figure 3: Command-space decision landscapes for four qualitative showcase scenarios. The title explains the flip, which return was chosen, and the timestep of the rollout. We mark the original command (circle) and the C&W (cross), our CF-ZOO (star), and nearest Pareto-front (diamond) counterfactuals, with arrows for the local margin gradient. Axes show true values in rewards.

Outcome	Cases	% all [95% CI]	% feasible [95% CI]
Success†	2515	78.7 [77.3,80.1]	**94.4** [93.5,95.2]
Grid-infeasible foil	531	16.6 [15.4,18.0]	–
Search-limited failure	149	4.7 [4.0,5.5]	5.6 [4.8,6.5]

Table 3: CF-ZOO diagnosis. Feasibility is assessed using a grid with 200 values per command dimension, yielding 200^d commands for $R \in \mathbb{R}^d$. Percentages in the % feasible column use only the feasible cases. †Includes 93 additional cases recovered by increasing the budget to 100k. Brackets denote 95% Wilson confidence intervals.

6 Discussion and Scope

The output should be interpreted locally. A successful counterfactual means that, at state s_t, the same trained PCN would prefer a_f under command R_{cf}. It does not imply that R_{cf} is Pareto-optimal. Conversely, failure may indicate optimizer failure, an unsupported foil, or a policy that is locally insensitive to the desired-return command.

Gajcin and Dusparic argue that RL counterfactuals cannot be imported directly from supervised learning because RL decisions are sequential and temporally embedded [Gajcin and Dusparic, 2024c]. RACCER addresses this by searching for an action sequence that reaches a counterfactual state where the desired action is likely under the policy [Gajcin and Dusparic, 2024b]. This answers the question of how an agent could reach another state in which it would choose the foil action. Our question is different. We ask why the PCN did not choose a_f at the original state s_t. Moving the agent to another state can be operationally reachable but explanatorily indirect. We therefore intervene on the PCN command rather than the environment state. Since the desired-return command is the user-facing trade-off input, R_{cf} is actionable by construction: it can be issued directly to the same policy at the same state. This does not guarantee that the resulting return is achievable, but it makes the counterfactual intervention itself directly implementable.

The seven desiderata therefore specialize differently in our setting [Gajcin and Dusparic, 2024c]. *Validity* is the margin condition $M_f(R_{cf}) \geq \kappa$, meaning the foil becomes preferred by the PCN. *Proximity* is the scaled command distance $\|D^{-1}(R_{cf} - R_t)\|_2$, which keeps the requested trade-off close to the original command. *Actionability* is command actionability: the user can directly issue the counterfactual command. *Sparsity* corresponds to changing few objective dimensions, although our current objective emphasizes small scaled distance rather than an explicit ℓ_0 penalty. In practice, the explanations are usually sparse, changing 1 or 2 entries. *Data-manifold closeness* is approximated by restricting commands to feasible bounds and using logged Pareto-front commands as behavioural anchors. *Causality* is local and interventional: the state and trained model are held fixed while only the command is changed. *Recourse* is immediate: replacing the directly settable command R_t with R_{cf} realizes the counterfactual.

7 Conclusion and Future Work

This paper introduces desired-return counterfactual explanations for command-conditioned MORL policies, such as PCNs, and proposes CF-ZOO to compute them. For a fixed policy and state, each explanation identifies how the requested trade-off would need to change to induce a specified foil action. Although currently PCN-specific, the approach could extend to other MORL or goal-conditioned agents with user-controllable preference, goal, or utility-conditioning inputs, such as PD-MORL agents [Basaklar *et al.*, 2023].

The method also has limitations. Failures may reflect infeasible foils or optimizer failure, as shown in Section 5.3.

Human studies are also needed, especially with participants outside MORL and RL, to test whether these explanations are understandable and useful. Prior RL-counterfactual work has evaluated whether non-expert users can identify flawed agents from counterfactual explanations, and whether counterfactual action-outcome visualizations improve users' understanding of agent preferences [Olson *et al.*, 2021; Amitai *et al.*, 2024]. A similar study could test whether people can use command counterfactuals to distinguish well-trained agents from poorly trained agents in the same environment.

Acknowledgments

This research has received funding from the project ALIGN4Energy (NWA.1389.20.251) of the research programme NWA ORC 2020 which is (partly) financed by the Dutch Research Council (NWO), and from the project PEER (grant agreement number 101120406) in the European Union's Horizon Europe Research and Innovation Programme.

References

[Abels *et al.*, 2019] Axel Abels, Diederik M. Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. Dynamic weights in multi-objective deep reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 11–20. PMLR, 2019.

[Amitai *et al.*, 2024] Yotam Amitai, Yael Septon, and Ofra Amir. Explaining reinforcement learning agents through counterfactual action outcomes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 10003–10011, 2024.

[Barrett and Narayanan, 2008] Leon Barrett and Srini Narayanan. Learning all optimal policies with multiple criteria. In *Proceedings of the 25th International Conference on Machine Learning*, pages 41–47. ACM, 2008.

[Basaklar *et al.*, 2023] Toygun Basaklar, Suat Gumussoy, and Ümit Y. Ogras. PD-MORL: Preference-driven multi-objective reinforcement learning algorithm. In *International Conference on Learning Representations*, 2023.

[Carlini and Wagner, 2017] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*, pages 39–57. IEEE Computer Society, 2017.

[Chen *et al.*, 2017] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

[Cheng *et al.*, 2019] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Query-efficient hard-label black-box attack: An optimization-based approach. In *International Conference on Learning Representations*, 2019.

[Dandl *et al.*, 2020] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In *Parallel Problem Solving from Nature – PPSN XVI*, volume 12269 of *Lecture Notes in Computer Science*, pages 448–469. Springer, 2020.

[Dazeley *et al.*, 2023] Richard Dazeley, Peter Vamplew, and Francisco Cruz. Explainable reinforcement learning for broad-xai: A conceptual framework and survey. *Neural Computing and Applications*, 35(23):16893–16916, 2023.

[Deshmukh *et al.*, 2023] Shripad V. Deshmukh, Srivatsan R, Supriti Vijay, Jayakumar Subramanian, and Chirag Agarwal. Counterfactual explanation policies in RL, 2023. arXiv:2307.13192; presented at the ICML 2023 Workshop on Counterfactuals in Minds and Machines.

[Felten *et al.*, 2023] Florian Felten, Lucas N. Alegre, Ann Nowé, Ana L. C. Bazzan, El-Ghazali Talbi, Grégoire Danoy, and Bruno C. da Silva. A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. Datasets and Benchmarks Track.

[Gajcin and Dusparic, 2024a] Jasmina Gajcin and Ivana Dusparic. ACTER: Diverse and actionable counterfactual sequences for explaining and diagnosing RL policies, 2024. arXiv:2402.06503.

[Gajcin and Dusparic, 2024b] Jasmina Gajcin and Ivana Dusparic. RACCER: Towards reachable and certain counterfactual explanations for reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 632–640. International Foundation for Autonomous Agents and Multiagent Systems, 2024.

[Gajcin and Dusparic, 2024c] Jasmina Gajcin and Ivana Dusparic. Redefining counterfactual explanations for reinforcement learning: Overview, challenges and opportunities. *ACM Computing Surveys*, 56(9):219:1–219:33, 2024.

[Hayes *et al.*, 2022] Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.

[Huber *et al.*, 2023] Tobias Huber, Maximilian Demmler, Silvan Mertes, Matthew L. Olson, and Elisabeth André. GANterfactual-RL: Understanding reinforcement learning agents' strategies through visual counterfactual explanations. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems*, pages 1097–1106. International Foundation for Autonomous Agents and Multiagent Systems, 2023.

[Milani *et al.*, 2024] Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. Explainable reinforcement learning: A survey and comparative review. *ACM Computing Surveys*, 56(7):168:1–168:36, 2024.

[Mothilal *et al.*, 2020] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 607–617. ACM, 2020.

[Nicolae *et al.*, 2018] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Mar-

tin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian M. Molloy, and Ben Edwards. Adversarial robustness toolbox v1.0.0. *CoRR*, abs/1807.01069, 2018. arXiv:1807.01069.

[Olson *et al.*, 2021] Matthew L. Olson, Roli Khanna, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. Counterfactual state explanations for reinforcement learning agents via generative deep learning. *Artificial Intelligence*, 295:103455, 2021.

[Piazza *et al.*, 2021] Nancirose Piazza, Yaser Faghan, Vahid Behzadan, and Ali Fathi. Adversarial attacks on deep algorithmic trading policies. In *Proceedings of the Conference on Applied Machine Learning in Information Security*, volume 3095 of *CEUR Workshop Proceedings*, pages 70–83. CEUR-WS.org, 2021.

[Reymond *et al.*, 2022] Mathieu Reymond, Eugenio Bargiacchi, and Ann Nowé. Pareto conditioned networks. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1110–1118. International Foundation for Autonomous Agents and Multiagent Systems, 2022.

[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.

[Samadi *et al.*, 2024] Amir Samadi, Konstantinos Koufos, Kurt Debattista, and Mehrdad Dianati. SAFE-RL: Saliency-aware counterfactual explainer for deep reinforcement learning policies. *IEEE Robotics and Automation Letters*, 9(11):9994–10001, 2024.

[Sutton and Barto, 2018] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2 edition, 2018.

[Tsirtsis *et al.*, 2021] Stratis Tsirtsis, Abir De, and Manuel Gomez-Rodriguez. Counterfactual explanations in sequential decision making under uncertainty. In *Advances in Neural Information Processing Systems*, volume 34, pages 30127–30139. Curran Associates, Inc., 2021.

[Vamplew *et al.*, 2011] Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84(1–2):51–80, 2011.

[Vamplew *et al.*, 2021] Peter Vamplew, Cameron Foale, Richard Dazeley, and Adam Bignold. Potential-based multiobjective reinforcement learning approaches to low-impact agents for AI safety. *Engineering Applications of Artificial Intelligence*, 100:104186, 2021.

[Verma *et al.*, 2024] Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan E. Hines, John P. Dickerson, and Chirag Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *ACM Computing Surveys*, 56(12):312:1–312:42, 2024.

[Wachter *et al.*, 2018] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31(2):841–887, 2018.

[Yang *et al.*, 2019] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Advances in Neural Information Processing Systems*, volume 32, pages 14610–14621. Curran Associates, Inc., 2019.

[Zhang *et al.*, 2023] Dayu Zhang, Nasser Lashgarian Azad, Sebastian Fischmeister, and Stefan Marksteiner. Zeroth-order optimization attacks on deep reinforcement learning-based lane changing algorithms for autonomous vehicles. In *Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics*, volume 1, pages 665–673. SCITEPRESS, 2023.

A Appendix

The Appendix is organized as follows:

- Implementation Details
- CF-ZOO Hyperparameters
- Return-Only PCN Adjustment
- Landscape and Grid Construction
- Runtime Details

A.1 Implementation Details

Scaled Command Coordinates

Because reward objectives can have different ranges and units, raw perturbation distances are not comparable. We therefore use scaled command coordinates. Let $R_{\min}$ and $R_{\max}$ be feasible command bounds and define

$$D = \mathrm{diag}(\sigma), \qquad \sigma = R_{\max} - R_{\min}. \tag{15}$$

Invalid or near-zero ranges are replaced by 1.

Given a raw perturbation δ,

$$z = D^{-1}\delta, \qquad \delta = Dz. \tag{16}$$

Equivalently,

$$z_i = \frac{\delta_i}{R_{\max,i} - R_{\min,i}}. \tag{17}$$

The scaled distance is

$$\|z\|_2 = \left\|D^{-1}\delta\right\|_2 = \left(\sum_{i=1}^{d}\left(\frac{\delta_i}{R_{\max,i} - R_{\min,i}}\right)^2\right)^{1/2}. \tag{18}$$

Commands are projected to the feasible box

$$\mathcal{C} = \{R : R_{\min} \le R \le R_{\max}\}. \tag{19}$$

Thus, optimization evaluates

$$R = \Pi_{\mathcal{C}}(R_t + Dz). \tag{20}$$

Binary Refinement Along Successful Rays

A successful endpoint may be farther than necessary, so we refine it by searching for the smallest successful step:

$$\alpha_f(u) = \inf\{\alpha \in [0, \alpha_{\max}(u)] : M_f(R_t + \alpha D u) \geq \kappa\}. \tag{21}$$

Binary search starts with

$$\alpha_{\text{lo}} = 0, \qquad \alpha_{\text{hi}} = \alpha_{\max}(u).$$

At each step,

$$\alpha_{\text{mid}} = \frac{\alpha_{\text{lo}} + \alpha_{\text{hi}}}{2}$$

is evaluated. If $M_f(R_t + \alpha_{\text{mid}} D u) \geq \kappa$, set $\alpha_{\text{hi}} = \alpha_{\text{mid}}$; otherwise set $\alpha_{\text{lo}} = \alpha_{\text{mid}}$.

We use 16 refinement steps. After K steps, the remaining interval is at most 2^{-K} of the original interval. With $K = 16$,

$$2^{-16} \approx 1.5 \times 10^{-5}$$

of the original ray interval.

ZOO-ADAM Coordinate Optimizer

This optimizer is a re-implementation of the ZOO-ADAM update rule from the ZOO attack [Chen *et al.*, 2017].

The optimizer works in normalized coordinates:

$$z = D^{-1}\delta, \qquad \delta = Dz,$$

where $D = \text{diag}(R_{\max} - R_{\min})$. Each policy query uses

$$R = \Pi_{\mathcal{C}}(R_t + Dz).$$

At each iteration, coordinate $i \in \{1, \ldots, d\}$ is sampled uniformly and evaluated at

$$z^+ = z + he_i, \qquad z^- = z - he_i,$$

where e_i is the i-th basis vector. In raw coordinates,

$$\delta^+ = Dz^+, \qquad \delta^- = Dz^-.$$

The coordinate derivative is estimated by

$$\widehat{\nabla_i \mathcal{L}}(z) = \frac{\mathcal{L}(z + he_i) - \mathcal{L}(z - he_i)}{2h}. \tag{22}$$

We use $h = 10^{-3}$, requiring two policy queries.

The estimate updates coordinate-wise ADAM:

$$m_i^{(k)} = \beta_1 m_i^{(k-1)} + (1 - \beta_1)\widehat{\nabla_i \mathcal{L}}(z^{(k)}), \tag{23}$$

$$v_i^{(k)} = \beta_2 v_i^{(k-1)} + (1 - \beta_2)\left(\widehat{\nabla_i \mathcal{L}}(z^{(k)})\right)^2. \tag{24}$$

We use

$$\beta_1 = 0.9, \qquad \beta_2 = 0.999, \qquad \epsilon = 10^{-8}.$$

Bias correction uses t_i, the number of updates to coordinate i:

$$\hat{m}_i^{(k)} = \frac{m_i^{(k)}}{1 - \beta_1^{t_i}}, \qquad \hat{v}_i^{(k)} = \frac{v_i^{(k)}}{1 - \beta_2^{t_i}}. \tag{25}$$

The coordinate update is

$$z_i^{(k+1)} = z_i^{(k)} - \eta \frac{\hat{m}_i^{(k)}}{\sqrt{\hat{v}_i^{(k)}} + \epsilon}, \tag{26}$$

with $\eta = 0.01$. Other coordinates are unchanged.

After the update,

$$\delta^{(k+1)} = Dz^{(k+1)},$$

and the PCN is queried at

$$R^{(k+1)} = \Pi_{\mathcal{C}}(R_t + \delta^{(k+1)}).$$

The method returns the closest valid command found under

$$\left\| D^{-1}(R_{cf} - R_t) \right\|_2.$$

A.2 CF-ZOO Hyperparameters

Table 4 gives the CF-ZOO settings.

Parameter	White-box comparison	Stress test
Random directions	21,000	9,001
Maximum queries	21,000	9,001
Ray binary-search steps	16	16
Margin κ	0.05	0.05
Penalty coefficient c	1.0	1.0
ZOO-ADAM learning rate	0.01	0.01
Finite-difference step h	10^{-3}	10^{-3}
Random seed	0	0

Table 4: CF-ZOO hyperparameters used in the reported experiments. The white-box comparison uses the same settings as the stress test except for the random-direction and query budgets.

A.3 Return-Only PCN Adjustment

The original PCN formulation conditions the policy on both a desired return and a desired horizon. In our experiments, horizon conditioning sometimes introduced additional variation into the command-action mapping rather than consistently producing reliable temporal control. We therefore removed the horizon from the policy input, but retained its useful effect indirectly through trajectory filtering and relabelling.

First, when multiple trajectories achieved the same return vector, we preferred the shorter trajectory. This biases the replay buffer toward more efficient demonstrations for the same achieved outcome. Second, before relabelling trajectories, we removed reward-neutral suffixes. These are final parts of trajectories in which the agent no longer receives additional reward. Trimming them prevents unnecessary waiting or wandering behaviour from being treated as part of the intended command-conditioned behaviour.

The return-only and return–horizon variants produced comparable evaluation results and both learned useful multi-objective behaviours. The return-only variant was therefore selected because it preserved empirical performance while simplifying the command space.

A.4 Landscape and Grid Construction

Each panel in Figure 3 shows how the policy's decision changes when two components of the desired-return command are varied. Everything else is kept fixed: the state s_t, the foil action a_f, the trained policy, and all command components that are not shown on the two axes.

Let p and q be the two displayed command dimensions. We select 121 equally spaced values for each dimension and evaluate every possible pair. This produces

$$121^2 = 14{,}641$$

commands for each panel.

The policy is queried once for every command on this grid. We then calculate the foil margin $M_f(R)$. The colors in each panel show the value of this margin, while the contour

$$M_f(R) = \kappa$$

marks the required validity threshold. Commands satisfying $M_f(R) \geq \kappa$ make the foil sufficiently preferred and are therefore valid counterfactual commands.

Only two command dimensions can be displayed at once. Each panel is therefore a two-dimensional slice through the full d-dimensional command space. It shows what happens when the two displayed dimensions change while all remaining dimensions stay fixed at their original values. It should not be interpreted as a complete visualization of the full command space.

The feasibility audit in Table 3 uses a different, higher-dimensional grid. Instead of varying only two displayed dimensions, it varies all d command dimensions. We use 200 equally spaced values for each dimension and take their Cartesian product:

$$\mathcal{G} = \mathcal{G}_1 \times \cdots \times \mathcal{G}_d, \qquad |\mathcal{G}| = 200^d.$$

Thus, a two-dimensional command has $200^2 = 40{,}000$ grid points, while a three-dimensional command has $200^3 = 8{,}000{,}000$ grid points.

A case is classified as grid-feasible if at least one evaluated command satisfies

$$\exists R \in \mathcal{G} \quad \text{such that} \quad M_f(R) \geq \kappa.$$

This means that the grid search found at least one command that makes the foil action valid.

A.5 Runtime Details

Table 5 reports mean wall-clock time per counterfactual method. CF-ZOO is slower because it adds directional screening, refinement, and zeroth-order local search.

Env.	Method	Time (s)
Branch	WHITE-CW	9.44
Branch	CF-ZOO	53.21
B-Bottles	WHITE-CW	8.55
B-Bottles	CF-ZOO	53.08
Collect	WHITE-CW	7.98
Collect	CF-ZOO	45.38
DST	WHITE-CW	8.03
DST	CF-ZOO	48.11
Fruit	WHITE-CW	10.21
Fruit	CF-ZOO	56.48
Minecart	WHITE-CW	8.63
Minecart	CF-ZOO	48.71
R-Gather	WHITE-CW	11.37
R-Gather	CF-ZOO	41.28
R-Line	WHITE-CW	9.07
R-Line	CF-ZOO	47.49
All	WHITE-CW	8.73
All	CF-ZOO	48.78

Table 5: Mean runtime per counterfactual query. The aggregate row is weighted by the number of cases in each environment. Environment abbreviations follow the notation used in the main paper.

Table 6 reports mean and median wall-clock time per query across budgets. Runtime grows steadily with the budget. We recommend using 50k for the best compromise between results and time.

Absolute runtimes are hardware- and implementation-dependent and may be reduced through faster hardware or further code optimization.

Budget	Mean $\pm$ std (s)	Median (s)
10k	19.1 ± 0.8	19.0
20k	43.7 ± 1.8	43.5
30k	60.5 ± 2.5	60.4
50k	101.4 ± 2.6	102.1
75k	108.4 ± 3.5	107.3
100k	176.0 ± 6.4	178.0

Table 6: Mean ($\pm$ standard deviation) and median wall-clock time per counterfactual query across query budgets.